ВСЕСОЮЗНЫЙ
НАУЧНО-ИССЛЕДОВАТЕЛЬСКИЙ ИНСТИТУТ
ИНФОРМАЦИИ И ЭКОНОМИКИ МИНПРИБОРА СССР
(ИНФОРМПРИБОР)

КАТАЛОГ

КОМПЛЕКСЫ ТЕХНИЧЕСКИХ СРЕДСТВ

Средства вычислительной техники

ВЫЧИСЛИТЕЛЬНЫЙ КОМПЛЕКС СМ 1700

ЕСЭТ

Москва 1988

УДК 681.3

Г. А. Егоров, В. В. Родионов

ИНФОРМПРИБОР выпускает третье издание каталога Государственной системы промышленных приборов и средств автоматизации (ГСП) под общей редакцией канд. техн. наук В. А. Рухадзе.

Каталог издается в виде отдельных томов. Каждый том состоит из нескольких выпусков, содержащих описание технических средств ГСП, объединенных по отдельным измеряемым физическим величинам или выполняемым функциям в составе АСУТП.

Структура каталога включает следующие тома:

Том 1. «Общее описание ГСП».

Том 2. «Средства получения информации о параметрах технологических процессов».

Том 3. «Средства локального контроля и автоматизации».

Том 4. «Средства централизованного контроля и регулирования».

Том 5. «Средства вычислительной техники».

Том 6. «Средства воздействия на процесс».

Том 7. «Типовые конструкции и элементы».

Материалы каталога предназначены для проектировщиков автоматизированных систем управления технологическими процессами, разработчиков средств автоматизации контроля и служб эксплуатации этих средств.

В настоящем выпуске приведено общее описание вычислительного комплекса СМ 1700, его назначение, области применения, рассмотрены архитектурные особенности и приведены краткие описания и показатели технических и программных средств, входящих в ВК СМ 1700.

По вопросам, касающимся издания каталога, просим обращаться по адресу: 125877, ГСП, Москва, А-252, Чапаевский пер., 14, ИНФОРМПРИБОР.

Ответственный за выпуск И. Н. Морозова

МИНИСТЕРСТВО ПРИБОРОСТРОЕНИЯ, СРЕДСТВ АВТОМАТИЗАЦИИ И СИСТЕМ УПРАВЛЕНИЯ СССР

ИНФОРМПРИБОР

Комплексы технических средств
Отраслевой каталог

Средства вычислительной техники

ВЫЧИСЛИТЕЛЬНЫЙ КОМПЛЕКС СМ 1700

ГОСУДАРСТВЕННАЯ СИСТЕМА ПРОМЫШЛЕННЫХ ПРИБОРОВ
И СРЕДСТВ АВТОМАТИЗАЦИИ

Москва 1988

НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

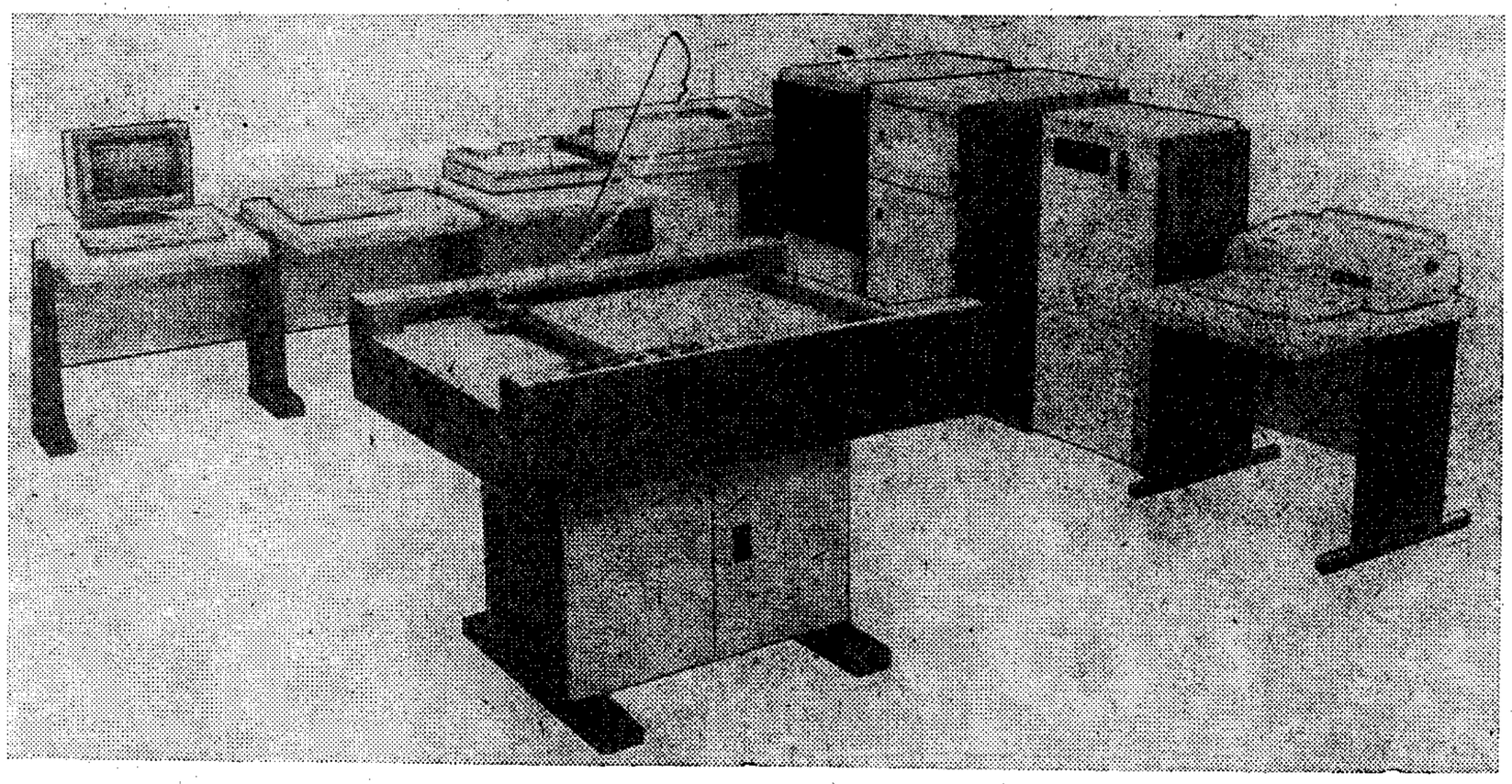

Рис. 1. Вычислительный комплекс СМ 1700

32-разрядная мини-ЭВМ СМ 1700 (рис. 1) предназначена для построения вычислительных комплексов, применяемых в системах автоматизации проектирования (САПР), в гибком автоматизированном производстве (ГАП), в системах, обрабатывающих планово-экономическую и учетно-статистическую информацию, в системах автоматизации комплексных научных исследований, в автоматизированных системах управления предприятиями (АСУП), в информационно-справочных и обучающих системах.

СМ 1700 является эволюционным развитием широко распространенных 16-разрядных мини-ЭВМ СМ-4, СМ 1420, СМ 1600 и др. с системным интерфейсом «Общая шина» (ОШ). Основной целью создания 32-разрядной модели СМ ЭВМ является потребность увеличить размер виртуального адресного пространства, так как этот показатель является одним из главных, определяющих общую производительность вычислительной системы.

СТРУКТУРА И СОСТАВ

Структура комплекса СМ 1700 приведена на рис. 2. Центральная часть и контроллеры конструктивно располагаются в одном монтажном блоке:

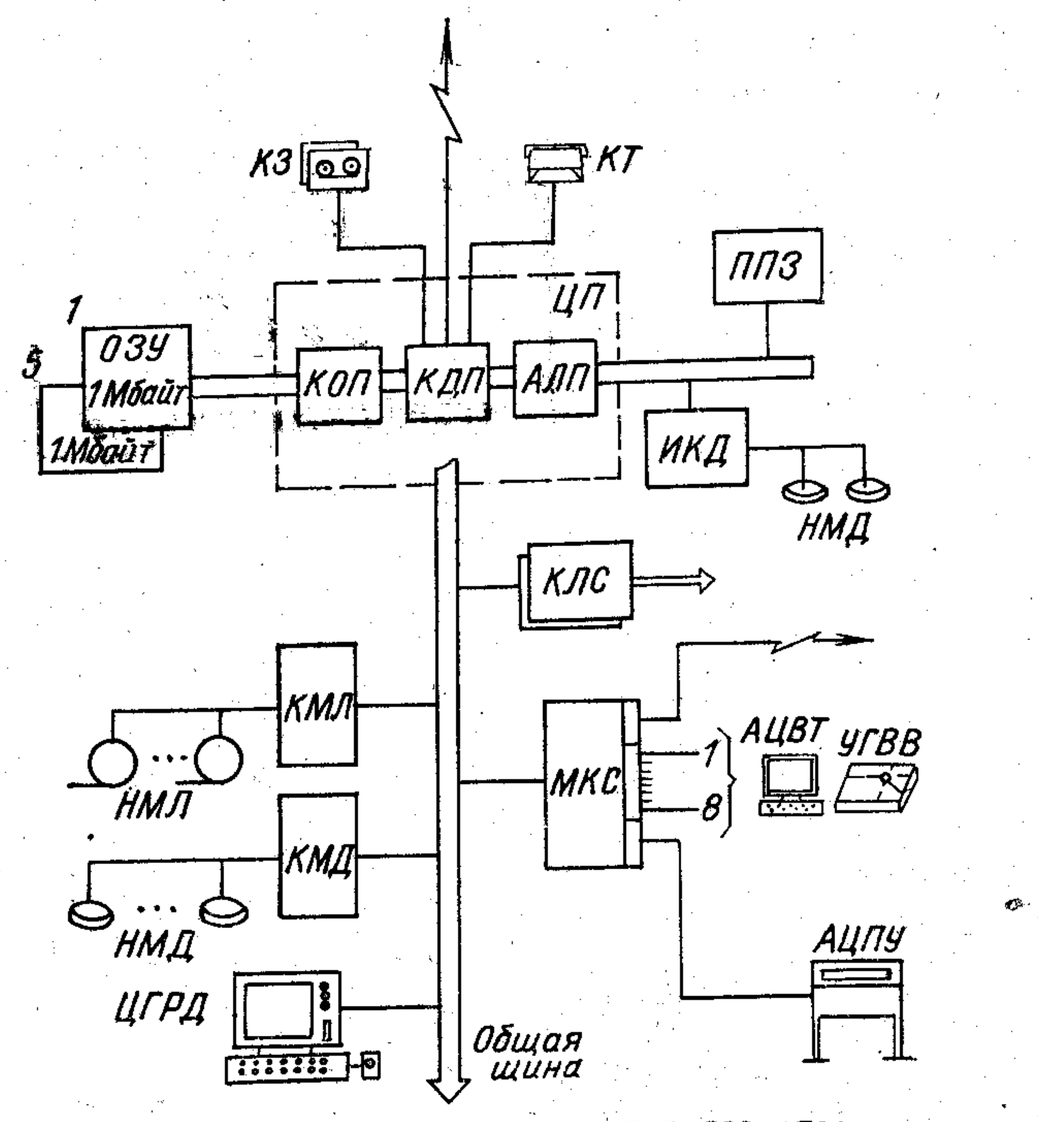

Рис. 2. Структурная схема ВК СМ 1700

центральный процессор (ЦП); арифметико-логический процессор (АЛП); консольно-диагностический процессор (КДП); контроллер оперативной памяти, (КОП); оперативное запоминающее устройство (ОЗУ); процессор с плавающей запятой (ППЗ); интегрированный контроллер дисков (ИКД); многофункциональный контроллер связи (МКС); контроллер локальной сети (КЛС); контроллер магнитных лент (КМЛ); контроллер магнитных дисков (КМД).

Все перечисленные аппаратные средства выполнены в виде одноплатных модулей размером 411,15×220 мм, за исключением КЛС (две платы). Каждая многослойная печатная плата (4...6 слоев) имеет четыре разъема СНП-59/64, с помощью которых подключается к объединительной многослойной (12 слоев) печатной плате в стойке вычислительной машины. Используются источники питания, обеспечивающие следующие показатели +5 В/100 А, +12 В/3 А, —15 В/3 А.

К указанным выше логическим модулям подсоединяются следующие устройства: консольный терминал (КТ); консольный загрузчик (КЗ); накопители на магнитных дисках (НМД); накопители на магнитных лентах (НМЛ); алфавитно-цифровые видеотерминалы (АЦВТ); устройства графического ввода-вывода (УГВВ); алфавитно-цифровое печатающее устройство (АЦПУ):

Непосредственно к ОШ подсоединяется цветной графический растровый дисплей (ЦГРД).

Устройства и модули, входящие в состав СМ 1700, встраиваются в стойки или размещаются на столах (подставках).

Комплектность поставки СМ 1700 приведена в табл. 1.

Таблица 1

Наименование и обозначение составных частей	Типовой комплекс СМ 1700														Изготовитель
	01	02	03	04	05	06	07	08	09	10	11	12	13	14	
Машина вычислительная СМ 2700 в составе:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Литовское ПО «Сигма»
арифметико-логического процессора СМ 2700.2400	1	1	1	1	1	1	1	1	1	1	1	1	1	1	То же
консольно-диагностического процессора СМ 2700.2805	1	1	1	1	1	1	1	1	1	1	1	1	1	1	»
контроллера оперативной памяти СМ 2700.2007	1	1	1	1	1	1	1	1	1	1	1	1	1	1	»
модуля ОЗУ — 1 Мбайт СМ 1700.3522	1	1	1	1	1	1	1	1	1	1	1	1	1	1	»
процессора с плавающей запятой СМ 2700.2008	1	1	1	1	1	1	1	1	1	1	1	1	1	1	»
многофункционального контроллера связи СМ 1700.4304 (эмуляция *DMF*32)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	»
консольного загрузчика СМ 5218 (эмуляция *TU*58)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Одесское ПО «Электронмаш»

Окончание табл. 1

Наименование и обозначение составных частей	Типовой комплекс СМ 1700														Изготовитель
	01	02	03	04	05	06	07	08	09	10	11	12	13	14	
консольного терминала СМ 6380 (эмуляция *LA*120)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Орловский завод УВМ им. К. Н. Руднева
заглушки ОШ	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Литовское ПО «Сигма»
инженерного пульта	1	1	1	1	1	1	1	1	1	1	1	1	1	1	То же
стойки 1200×800×400	1	1	1	1	1	1	1	1	1	1	1	1	1	1	»
стола	1	1	1	1	1	1	1	1	1	1	1	1	1	1	»
Модуль ОЗУ — 1 Мбайт СМ 1700.3522	1	1	1	1	1	3	4	3	1	2	1	2	2	1	»
Многофункциональный контроллер связи СМ 1700.4304 (эмуляция *DMF*-32)	—	—	—	1	—	1	1	—	—	—	—	1	1	1	»
Устройство внешней памяти СМ 1700.5309	—	—	1	—	—	1	1	1	1	1	1	1	1	1	»
(НМЛ СМ 5309, 40 Мбайт, 32 и 63 бит/мм; эмуляция *TS*11 или *TU*80)	—	—	1	—	—	1	1	1	1	1	1	1	1	1	«Изот» (НРБ)
Устройство внешней памяти СМ 1700.5408	1	—	1	1	1	1	1	1	—	1	1	—	1	—	Литовское ПО «Сигма»
(НМД СМ 5408, 14 Мбайт съемный; эмуляция *RK*06)	2	—	2	5	2	5	2	2	—	2	2	—	2	—	То же
Устройство внешней памяти СМ 1700.5514	—	1	—	—	1	1	—	—	1	—	1	1	—	1	»
(НДМ СМ 5514, 23,4 Мбайт несъемный; эмуляция *RK*07)	—	3	—	—	3	3	—	—	3	—	3	3	—	3	»
Устройство внешней памяти СМ 1700.5504	—	—	1	—	—	—	—	1	1	—	1	1	1	—	»
(НМД СМ 5504, 121 Мбайт несъемный; эмуляция *RM*80 или *R*80)	—	—	1	—	—	—	—	2	2	—	2	1	2	—	«Роботрон» (ГДР)
Устройство внешней памяти на кассетной магнитной ленте СМ 5219	—	1	—	—	—	—	—	—	—	—	—	—	—	—	Одесское ПО «Электронмаш»
Расширитель интерфейса ОШ	1	—	1	1	1	1	1	1	—	1	1	—	1	—	Литовское ПО «Сигма»
Видеотерминал алфавитно-цифровой:															
СМ 7238.00 (эмуляция *VT*220)	1	1	4	12	1	4	4	1	1	2	2	4	—	2	Винницкое ПО «Терминал»
СМ 7238.01 с граф. возм. (эмуляция *VT*240)	—	—	—	—	—	—	4	3	1	2	—	6	8	2	То же
Дисплей растровый графический цветной СМ 7317 (1024×768 точек; эмуляция *VS*11)	—	1	—	—	1	1	—	1	—	—	—	—	—	—	Литовское ПО «Сигма»
Устройство ввода графической информации (планшетного типа):															
СМ 6424.03 (280×280 мм)	—	1	—	1	1	1	—	1	—	—	—	1	—	1	Московское НПО «Оргтехника»
СМ 6423 (840×594 мм)	—	—	—	—	—	—	1	1	—	—	—	—	—	—	То же
Устройство вывода графической информации (планшетного типа):															
СМ 6470.02 (210×297 мм)	—	1	—	—	—	—	—	1	—	—	—	1	—	1	Одесское ПО «Электронмаш»
СМ 6408.02 (840×594 мм)	—	—	—	1	1	1	1	1	—	—	—	—	—	—	ПО «Краснодарский ЗИП»
Устройство печати растровое СМ 6334 (300 строк/мин; эмуляция *LXY*-11). Допускается замена на СМ 6361 («Видеотон», ВНР)	—	—	1	1	—	1	1	1	—	1	—	1	1	1	Орловский завод УВМ им. К. Н. Руднева
Устройство печати последовательное СМ 6308.01 180 знаков/с (твердая копия с экрана АЦВТ)	—	—	—	—	—	—	—	—	—	—	—	6	6	4	То же
Стол	1	3	4	13	2	5	8	6	2	4	2	11	8	6	Литовское ПО «Сигма»
Стойка 1200×800×800	1	—	2	2	1	3	3	3	1	3	2	1	3	1	То же
Операционная система (МОС ВП)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	»
Программная диагностика (МСПД)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	»
Микродиагностика (СМПД)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	»

ТЕХНИЧЕСКИЕ ДАННЫЕ

Объем виртуального адресного пространства, Гбайт	4
Объем оперативной памяти, Мбайт	1...5
Количество машинных инструкций	304
Количество режимов адресации	14
Тип обрабатываемых данных и их форматы:	
двоичные числа с фиксированной запятой, бит	8; 16; 32; 64; 128
двоичные числа с плавающей запятой (разрядность=мантисса+порядок), бит	32; 64; 128
десятичные числа, байт	0...16 (до 31 десятичной цифры)
символьные строки, байт	0...65535
битовые поля, бит	0...32
Производительность комплекса (на смеси «Ветстоун»), тыс. операций/с	300

Типы используемых интерфейсов	ОШ, ИРПР, ИРПС, С2, С2-ИС, СМД, ИМД-М, ИМЛ-П
Число уровней прерываний:	
программно-генерируемых	15
аппаратных	16
Операционные системы	МОС ВП, ДЕМОС-32
Языки программирования	Макроассемблер, ФОРТРАН, СИ, КОБОЛ, ПАСКАЛЬ, ПЛ-1, БЕЙСИК, Модула-2, Корал, Блисс-32, АДА, ЛИСП, Пролог
Язык микропрограммирования	МИАСС СМ
Габаритные размеры, мм:	
стойки вычислительной машины	1200×600×400
стойки ВЗУ	1200×600×800
стола	725×800×600
Условия эксплуатации:	
температура окружающего воздуха, °С	5...40
относительная влажность воздуха при 30° С, %	40...90
атмосферное давление, кПа	84...107

Показатели	Типовой комплекс СМ 1700													
	01	02	03	04	05	06	07	08	09	10	11	12	13	14
Потребляемая мощность, кВА	2,7	2,3	4,8	5,4	3,7	5,8	7,4	4,5	3,1	5,4	3,6	7,5	7,2	5,3
Масса, кг	606	510	1355	2085	1045	1833	1905	1734	522	1239	835	1251	1336	774
Занимаемая площадь, м²	12	15	25	39	15	38	42	43	15	25	15	46	43	33
Наработка на отказ, ч	3200	2300	2800	1400	1200	1100	1300	1100	3000	2600	2400	1400	1700	1800
Наработка на сбой, ч	320	230	280	140	120	110	130	110	300	260	240	140	170	180

ОСНОВНЫЕ АРХИТЕКТУРНЫЕ ПОЛОЖЕНИЯ

Архитектура ВК СМ 1700 в первую очередь призвана обеспечить эффективное выполнение машинных программ, генерируемых трансляторами с различных языков программирования, под управлением мощной операционной системы с виртуальной памятью.

Для достижения этой цели помимо ортогональности системы инструкций (коды операций, режимы адресаций и типы данных рассматриваются трансляторами независимо) широко используется аппаратная реализация целиком некоторых операторов языков высокого уровня и примитивов операционных систем. Это относится, например, к способу вызова подпрограмм, реализации сложных переходов на многие направления, управлению циклами, вычислению индексов массива, обработке очередей и списков, обработке контекста процесса, обслуживанию прерываний, обработке «семафоров» взаимодействующих процессов, преобразованию виртуального адреса, защите памяти и др.

Форматы инструкций и данных

В СМ 1700 используется переменный формат инструкций. Машинная инструкция состоит из кода операции, за которым следуют спецификации операндов. Для часто используемых инструкций код операции занимает 1 байт, в редко используемых — 2 байт. Количество операндов находится в прямой зависимости от кода операции и может колебаться от 0 до 6. Для спецификации одного операнда может потребоваться от 1 до 17 байт: первые 1 или 2 байт — собственно спецификатор операнда (определяет режим адресации операнда), за которым может следовать до 16 байт расширения спецификатора (смещение, адрес или непосредственно операнд).

Практически длина инструкции варьируется от 1 до 37 байт и может начинаться с любого адресуемого байта оперативной памяти, т. е. не требуется выравнивания на определенную границу памяти. Данные любого формата также могут располагаться в памяти по любому адресу. Этот принцип позволяет избежать неиспользуемых ячеек памяти.

ВК СМ 1700 имеет пять типов данных (целый, вещественный, битовый, десятичный и символьный), каждый может иметь несколько форматов.

Целый тип данных используется для представления чисел с фиксированной запятой (знаковых или беззнаковых) и имеет следующие форматы: *B* — байт (8 разрядов); *W* — слово (16 разрядов); *L* — длинное или двойное слово (32 разряда); *Q* — квадрослово (64 разряда); *O* — октаслово (128 разрядов).

Вещественный тип данных используется для представления чисел с плавающей запятой и имеет четыре формата: *F* — одинарный формат (32 разряда = 24 разряда (мантисса + 8 разрядов порядок);

D — двойной формат (64 разряда=56 разрядов мантисса+8 разрядов порядок); *G* — двойной формат расширенного диапазона (64 разряда=53 разряда мантисса+11 разрядов порядок); *H* — учетверенный формат (128 разрядов=113 разрядов мантисса+15 разрядов порядок).

Мантисса представляется в прямом коде в нормализованном виде и поэтому старший разряд, который всегда равен 1, в оперативной памяти не хранится, а учитывается аппаратурой в процессе обработки. Знак числа входит в число разрядов, отведенных под мантиссу.

Порядок представляется в коде со смещением на 2^7 (для форматов *F* и *D*), 2^{10} (для формата *G*) или 2^{14} (для формата *H*). Такое представление получается путем прибавления соответствующего смещения к исходному порядку в дополнительном коде. Это позволяет избежать переполнения порядка при его переходе от минимального отрицательного к максимальному положительному значению, что упрощает аппаратуру.

Битовый формат используется для представления битовых полей переменной длины. Чтобы описать битовое поле, необходимо наличие трех атрибутов; *A* — базового адреса, указывающего на байт памяти, являющегося основой для определения местонахождения поля: *P* — позиции начального разряда поля по отношению к младшему (нулевому) разряду базового байта; *S* — размера поля в битах.

Позиция может принимать значение -2^{31}... ($2^{31}-1$) бит; размер поля — от 1 до 32 бит.

ВК СМ 1700 оперирует со следующими форматами представления десятичных чисел: упакованные десятичные строки и числовые строки с лидирующим знаком, с завершающим знаком — зонное кодирование; перфокарточное кодирование.

Упакованные десятичные строки представляют собой последовательность байтов в памяти и определяются двумя атрибутами: *A* — адресом первого байта и *L* — количеством десятичных цифр в строке. В каждом байте компонуются две цифры за исключением последнего байта, где младшая цифра компонуется вместе со знаком. Значение *L* должно быть в пределах 0...31 (строка нулевой длины интерпретируется как 0).

В числовых строках каждый байт содержит одну цифру или знак. Числовые строки с лидирующим знаком представляют собой последовательность байтов в памяти и определяются двумя атрибутами: *A* — адресом первого байта (содержит знак) и *L* — длиной строки (0...31) в цифрах. Старшая цифра размещается вслед за знаком. Для кодирования цифр и знаков используется код КОИ-8.

Числовые строки с завершающим знаком представляют собой последовательность байтов в памяти и определяются двумя атрибутами: *A* — адрес первого байта (старшая цифра) и *L* — длина строки (0...31). Все цифры, кроме младшей, представляются байтами в коде КОИ-8. Младшая цифра компонуется вместе со знаком в последнем байте числовой строки и имеет различное кодирование для зонного и перфокарточного форматов. Если строка представляет собой беззнаковое десятичное число, то последний байт содержит только младшую цифру в коде КОИ-8.

Символьный тип данных используется для представления алфавитно-цифровых символьных строк в коде КОИ-8, которые имеют два атрибута: адрес и длину (в байтах или символах). Длина строки может быть от 1 до 65535 символов.

Регистры общего назначения

Операнды инструкций часто либо располагаются в общих регистрах, либо адресуются через них. Эти 16 32-разрядных программно доступных регистра имеют номера от 0 до 15 (в десятичной системе). Регистры могут быть использованы для временного хранения, накапливания промежуточных результатов или в качестве регистров базы и индексных регистров. Базовый регистр содержит адрес базы такой структуры данных программ, как таблица, а индексный регистр содержит логическое смещение в структуре данных.

Учетверенное слово данных (форматы *Q*, *D*, *G*) обычно загружается в два соседних регистра. Например, при засылке числа с плавающей запятой двойной точности в регистр *R7*, фактически загружаются регистры *R7* и *R8*. При загрузке октаслова задействованными оказываются сразу четыре регистра.

Регистры *R*12—*R*15 имеют специальное назначение для многих инструкций и поэтому имеют специальные обозначения.

PC или *R*15 — счетчик программы (Program Counter), который содержит адрес очередного байта в потоке инструкций.

SP или *R*14 — указатель стека (Stack Pointer), который содержит адрес вершины стека, используемый при вызовах подпрограмм и процедур.

FP или *R*13 — указатель кадра (Frame Pointer), который содержит адрес структуры данных, загружаемой в стек и называемой кадром вызова, который применяется при вызовах процедур.

AP или *R*12 — указатель аргумента (Argument Pointer), который содержит адрес структуры данных, называемый списком аргументов, используемым при вызове процедур.

Регистры *R*0—*R*5 имеют специальное назначение для инструкций, работающих со строками символов, упакованными десятичными строками и для инструкций циклического контроля и вычисления полинома. Вышеуказанные инструкции используют эти регистры для хранения промежуточных результатов, причем по завершении инструкции в регистрах остаются результаты, которые программа может использовать как операнды.

Регистры специального назначения, кроме счетчика программы, можно применять для других целей. Счетчик программы не может быть использован как накапливающий регистр, регистр для промежуточного хранения или индексный регистр. Однако не рекомендуется использовать указатель стека, указатель аргумента или указатель кадра для иных целей, чем те, для которых они предназначены.

Расширенное слово состояния процессора

Регистр процессора, называемый расширенным словом состояния процессора (*PSL*), определяет состояние процессора в текущий момент времени.

Младшие 16 разрядов расширенного слова состояния процессора (*PSL*) представляют собой слово состояния программы (*PSW*), доступное пользовательскому процессу. Слово состояния программы, представленное на рис. 3, содержит два типа битовых полей — коды условий и флаги особых ситуаций.

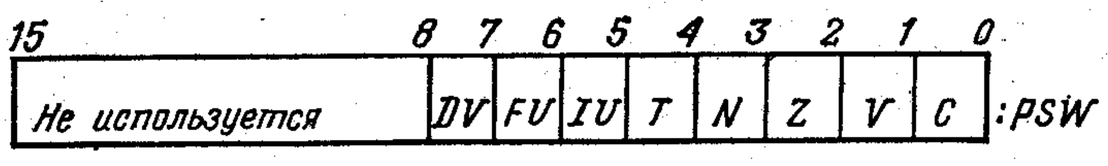

Рис. 3. Формат слова состояния программы *PSW*

DV — разрешение ловушки десятичного переполнения;
FU — разрешение ловушки потери значимости при операциях с плавающей запятой;
IV — разрешение ловушки целочисленного переполнения;
T — флаг трассы;
N — отрицательный результат;
Z — нулевой результат;
V — переполнение;
C — перенос (заем).

Коды условий *N*, *Z*, *V*, *C* определяются результатом выполнения очередной машинной инструкции.

Пользователь может эффективно и просто контролировать эти условия, либо проверяя коды условий, используя инструкции условных переходов, либо устанавливая флаги особых ситуаций. При установленных флагах процессор воспринимает переполнение целых чисел и десятичных строк, а также потерю значимости для плавающей запятой как особые ситуации.

Имеются два типа особых ситуаций, которые касаются пользовательского процесса: ошибки трассы (используются в отладчике) и арифметические особые ситуации.

Когда возникает особая ситуация, процессор немедленно сохраняет текущее состояние и передает управление операционной системе. Операционная система автоматически отыскивает процедуру, необходимую для обслуживания этой особой ситуации.

Формат слова состояния процессора *PSL* приведен на рис. 4.

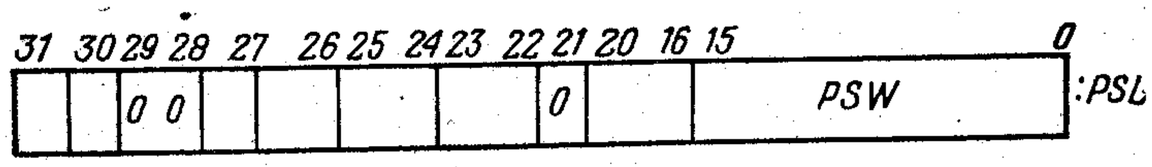

Рис. 4. Формат слова состояния процессора *PSL*

Старшие 16 разрядов *PSL* обеспечивают привилегированное управление системы.

Содержимое разрядов следующее:
31 — режим совместимости;
30 — разрешение трассы;
29, 28 — не используются;
27 — выполнена основная часть инструкции;
26 — работа со стеком прерывания;
25, 24 — текущий режим доступа;
23, 22 — предыдущий режим доступа;
20...16 — приоритет прерывания.

Набор инструкций, выполняемый процессором, определяется разрядом режима совместимости в расширенном слове состояния процессора. Этот разряд обычно устанавливается (режим совместимости) или сбрасывается (собственный режим) операционной системой.

Многие сложные инструкции (десятичная арифметика, обработка символьных строк и др.) имеют длительное время выполнения. Для таких инструкций в процессе их выполнения разрешены прерывания. И после обслуживания прерывания инструкция возобновляет свою работу. Если же инструкция достигает определенной точки, где должна выполняться запись результатов, то в *PSL* устанавливается разряд 27 и в дальнейшем инструкция не прерывается до своего завершения.

При многопрограммной работе процессор должен обеспечить защиту и разделение ресурсов системы между конкурирующими процессами. Основой для защиты в такой системе является режим доступа к процессору. В любой конкретный момент времени процессор либо выполняет инструкции в контексте определенного процесса, либо выполняет инструкции в общесистемном контексте обслуживания прерываний. В контексте процесса процессор различает четыре режима доступа: ядро (*K*), исполнитель (*E*), супервизор (*S*), пользователь (*U*). Режим ядра является наиболее привилегированным режимом, а режим пользователя — наименее привилегированным.

Большую часть своего времени процессор находится в режиме пользователя в контексте того или иного процесса. Когда программе пользователя необходимы обслуживающие программы операционной системы (либо для представления ресурсов и обслуживания ввода-вывода, либо для информации) он запрашивает эти обслуживающие программы.

Обслуживающие программы выполняются процессором в том же самом режиме доступа или в одном из более привилегированных режимов доступа внутри контекста этого процесса. Таким образом, все четыре режима доступа существуют внутри одного и того же виртуального адресного пространства. Каждый режим доступа имеет свой собственный стек в области управления процессорного пространства и поэтому каждый процесс имеет четыре стека: один для каждого режима доступа. Заметим, что это позволяет операционной системе легко осуществлять переключение контекста, даже во время выполнения обслуживания операционной системой.

Ни в каком режиме, кроме режима ядра, процессор не может выполнять такие инструкции, как останов процессора, загрузка и сохранение контекста процесса, доступ к внутренним регистрам процессора, обеспечивающим управление памятью, обслуживание прерываний, управление консольным терминалом, таймером или процессорными часами.

При любом режиме процессор не позволяет текущей инструкции иметь доступ к памяти, если этот режим не имеет привилегий на этот доступ. Возможность выполнять команды в одном из более привилегированных режимов предоставляется и контролируется операционной системой. Процессор реализует систему привилегированной защиты памяти: выполняемые инструкции одного режима могут защищать себя и любую часть своих структур данных от чтения и записи при выполнении инструкций в любом менее привилегированном режиме. Например, инструкции, выполняемые в режиме исполнителя, могут защищать свои структуры данных от инструкций, выполняемых в режиме супервизора или ядра. Инструкции, выполняемые в режиме супервизора, могут защищать свои структуры данных от доступа при выполнении инструкций в режиме пользователя. Такой механизм защиты памяти обеспечивает целостность структуры данных систем.

Режимы адресации

Режимы адресации процессора позволяют располагать операнды в регистре или памяти, или представлять непосредственно в теле инструкции в виде констант.

Базовые режимы адресации

Адресация операнда осуществляется с помощью так называемого «спецификатора операнда» (рис. 5), который располагается в теле машинной инструкции и занимает один байт (код режима адресации *M* и номер используемого общего регистра *Rn*).

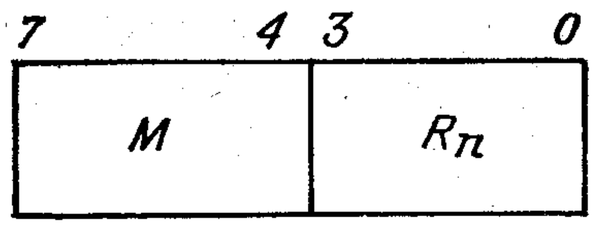

Рис. 5. Формат спецификатора операнда

Существует семь базовых режимов адресации, которые используют общие регистры для определения местонахождения операнда. Приведены мнемоника и краткое описание базовых режимов адресации:

Rn — регистровый режим, при котором регистр содержит операнд;

(*Rn*) — косвенно-регистровый режим, при котором регистр содержит адрес операнда;

— (*Rn*) — режим автоуменьшения, при котором содержимое регистра уменьшается на размер операнда, а затем используется как адрес операнда. Размер операнда в байтах, который определяется типом данных операнда и зависит от инструкции. Например, инструкция *CLRW* «Очистить слово» использует размер 2, так как в каждом слове содержится 2 байт;

(*Rn*) + — режим автоувеличения, при котором содержимое регистров используется как адрес операнда, а затем увеличивается на размер операнда. Если в качестве обозначенного регистра используется счетчик программы, то режим адресации называется непосредственным режимом (мнемоника: #K, где K — константа, представляющая собой расширение спецификатора длиной 1, 2, 4, 8 или 16 байт в зависимости от типа данных);

a (*Rn*) — режим косвенного автоувеличения, при котором содержимое регистра используется как адрес ячейки в памяти, содержащей адрес операнда и затем увеличивается на 4 (размер адреса). Если обозначенным регистром является счетчик программы, то режим называется абсолютным режимом (мнемоника: *a*#*A*, где *A* — адрес, являющийся расширением спецификатора длиной в 4 байт);

D (*Rn*) — режим смещения, при котором содержимое регистра используется как базовый адрес. Знаковая величина смещения *D* (расширение спецификатора длиной в байт, слово или двойное слово) добавляется к базовому адресу и полученная сумма представляет фактический адрес операнда. Количество байтов, занимаемых смещением, определяется автоматически на этапе трансляции. Если используемым регистром является счетчик программы, то режим называется относительным (мнемоника: *A*, где *A* — метка адресуемой ячейки);

aD (*Rn*) — режим косвенного смещения, при котором значение, содержащееся в регистре, используется как базовый адрес таблицы адресов. Знаковая величина смещения *D* (расширение спецификатора длиной в один байт, слово или двойное слово) добавляется к базовому адресу и полученная сумма является адресом ячейки, которая содержит действительный адрес операнда. Если в качестве регистра используется счетчик программы, то режим называется косвенно-относительный (мнемоника: *aA*, где *A* — метка ячейки памяти, содержащей адрес операнда).

Индексация режимов адресации

Из этих семи основных режимов все режимы, кроме регистрового режима, могут быть модифицированы с помощью индексного регистра, в качестве которого используется один из общих регистров. При этом название режима адресации состоит из имени основного режима с добавкой слов «с индексацией». В этом случае спецификатор операнда состоит из двух байт: первый байт содержит код режима индексации и номер индексного регистра, а второй байт — базовый режим адресации. Мнемоника также получается составной путем добавления обозначения [*Rx*], где *Rx* — индексный регистр. Например, индексный режим адресации для косвенно-регистрового режима называется «косвенно-регистровый с индексацией» и имеет мнемонику (*Rn*) [*Rx*].

При индексном режиме адресации регистр *Rn* используется для вычисления базового адреса структуры данных, а другой регистр *Rx* используется для вычисления индексного смещения внутри структуры данных. Чтобы получить фактический адрес операнда при индексном режиме адресации, процессор вычисляет базовый адрес операнда, зада-

ваемый одним из базовых режимов адресации (кроме регистрового режима), берет содержимое индексного регистра *Rx*, умножает его на размер, соответствующий типу данных операнда, и прибавляет полученное значение к базовому адресу операнда.

Литеральный режим адресации

Процессор также обеспечивает литеральный режим адресации, в котором старшие два разряда спецификатора операнда должны быть нулями, а беззнаковое шестибитное поле интерпретируется как константа (целое число *K* или число с плавающей запятой $M \times 2^F$ (рис. 6):

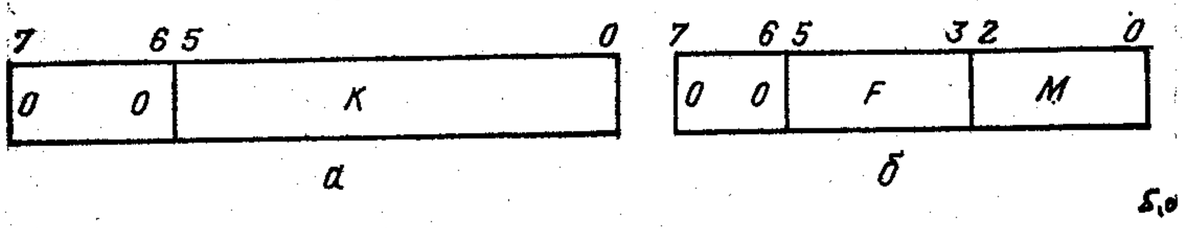

Рис. 6. Формат спецификатора операнда литерального режима адресации: а — для чисел с фиксированной запятой; б — для чисел с плавающей запятой

Режим литерала является очень эффективным для задания коротких констант и часто используется вместо непосредственного режима адресации. 6-разрядное поле интерпретируется как целое число, когда используется в операциях над целыми числами, и поэтому может определять константу *K* от 0 до +63 (в десятичной системе). 6-битное поле интерпретируется как константа с плавающей запятой, когда используется в операциях над числами с плавающей запятой, при которых 3 бит определяют порядок *F*, а 3 бит мантиссу *M*.

Таким образом, всего имеется 14 семантически различных режимов адресации (7 базовых+6 индексных+1 литеральный). Причем некоторые из них имеют несколько модификаций, например, режим смещения имеет три модификации в зависимости от длины смещения. В табл. 2 приведены краткие сведения по всем режимам адресации.

Список инструкций СМ 1700

В приведенном ниже перечне мнемоника некоторых инструкций содержит символ подчеркивания (_). Это означает, что данная инструкция имеет несколько модификаций в зависимости от типа данных, типа перехода, типа доступа, режима работы процессора и состояния отдельных битов. При программировании в мнемонике конкретной инструкции символ подчеркивания заменяется на одну из следующих букв:

тип данных: *B* — байт; *W* — слово (по умолчанию); *L* — удвоенное слово (длинное слово); *Q* — учетверенное слово (квадрослово); *O* — увосьмеренное слово (октаслово); *F* — одинарный формат (*F*-формат); *D* — двойной формат (*D*-формат); *G* — двойной формат расширенного диапазона (*G*-формат); *H* — учетверенный формат (*H*-формат);

режим процессора: *K* — ядро; *E* — исполнитель; *S* — супервизор; *U* — пользователь;

состояние бита: *C* — сброшен; *S* — установлен;

тип перехода: <пусто> — знаковый; *U* — беззнаковый;

тип доступа: *R* — чтение; *W* — запись.

Таблица 2

Код режима адресации	Регистры: *Rn*=*R*0—*R*14 Мнемоника	Наименование	*Rn*=*R*15 Мнемоника	Наименование	Возможность индексации
0—3	#*K*	Литеральный	Невозможно		Нет
4	[*Rx*]	Индексный	Недопустимо		Нет
5	*Rn*	Регистровый	Не рекомендуется		Нет
6	(*Rn*)	Косвенно-регистровый	Не рекомендуется		(*Rn*) [*Rx*]
7	—(*Rn*)	Автоуменьшение	Не рекомендуется		—(*Rn*) [*Rx*]
8	(*Rn*)+	Автоувеличение	#*K*	Непосредственный	(*Rn*)+[*Rx*] или #*K*[*Rx*]
9	*a*(*Rn*)+	Косвенное автоувеличение	#*aA*	Абсолютный	*a*(*Rn*)+[*Rx*] или *a*#*A*[*Rx*]
A,*C*,*E*	*D*(*Rn*)	Смещение	*A*	Относительный	*D*(*Rn*) [*Rx*] или *A* [*Rx*]
B,*D*,*F*	*aD*(*Rn*)	Косвенное смещение	*aA*	Косвенно-относительный	*aD* (*Rn*) [*Rx*] или *aDA* [*Rx*]

Примечания: 1. Мнемонику #K ассемблер автоматически транслирует в литеральный или непосредственный режим адресации в зависимости от величины константы K. 2. По мнемонике *D*(*Rn*) или A ассемблер автоматически создает код режима и смещение *D* требуемой длины (A-байт, C-слово, E-длинное слово). 3. По мнемонике *aD*(*Rn*) или *aA* ассемблер автоматически создает код режима и смещение *D* требуемой длины (*B*-байт, *D*-слово, *F*-длинное слово). 4. При режимах 5, 6, 7 не рекомендуется использовать *Rn*=*R*15, поскольку результат может быть непредсказуемым. 5. При индексации режимов 7, 8, 9 не рекомендуется использовать *Rn*=*Rx*, поскольку результат может быть непредсказуемым.

Логические инструкции для чисел с фиксированной и плавающей запятой

1. *MOV_* — пересылка (*B*, *W*, *L*, *F*, *D*, *H*, *G*, *O*, *Q*).
2. *MNEG_* — пересылка с отрицанием (*B*, *W*, *L*, *F*, *D*, *G*, *H*).
3. *MCOM_* — пересылка с инверсией (*B*, *W*, *L*).
4. *MOVZ_* — пересылка с лидирующими нулями (*BW*, *BL*, *WL*).
5. *CLR_* — очистка (*B*, *W*, *L=F*, *Q=D=g*, *O=H*).
6. *CVT__* — преобразование (*B*, *W*, *L*, *F*, *D*, *G*, *H*) в (*B*, *W*, *L*, *F*, *D*, *G*, *H*); кроме *BB*, *WW*, *LL*, *FF*, *HH*, *DG*, *GD*.
7. *CVTR_L* — преобразование (*F*, *D*, *G*, *H*) в длинное слово (с округлением).
8. *CMP_* — сравнение (*B*, *W*, *L*, *F*, *D*, *G*, *H*).
9. *TST_* — тест (*B*, *W*, *L*, *F*, *D*, *G*, *H*).
10. *BIS_2* — установ битов (*B*, *W*, *L*); 2 операнда.
11. *BIS_3* — установ битов (*B*, *W*, *L*); 3 операнда.
12. *BIC_2* — сброс битов (*B*, *W*, *L*); 2 операнда.
13. *BIC_3* — сброс битов (*B*, *W*, *L*); 3 операнда.
14. *BIT_* — тест битов (*B*, *W*, *L*).
15. *XOR_2* — исключающее «ИЛИ» (*B*, *W*, *L*); 2 операнда.
16. *XOR_3* — исключающее «ИЛИ» (*B*, *W*, *L*); 3 операнда.
17. *ROTL* — ротация длинного слова.
18. *PUSHL* — загрузка длинного слова в стек.

Арифметические инструкции для чисел с фиксированной и плавающей запятой

19. *INC_* — инкремент (*B*, *W*, *L*).
20. *DEC_* — декремент (*B*, *W*, *L*).
21. *ASH_* — арифметический сдвиг (*L*, *Q*).
22. *ADD_2* — сложение (*B*, *W*, *L*, *F*, *D*, *G*, *H*) 2 операнда.
23. *ADD_3* — сложение (*B*, *W*, *L*, *F*, *D*, *G*, *H*) 3 операнда.
24. *ADWC* — сложение с переносом.
25. *ADAWI* — сложение выровненных слов (с блокировкой памяти).
26. *SUB_2* — вычитание (*B*, *W*, *L*, *F*, *D*, *G*, *H*); 2 операнда.
27. *SUB_3* — вычитание (*B*, *W*, *L*, *F*, *D*, *G*, *H*); 3 операнда.
28. *SBWC* — вычитание с переносом.
29. *MUL_2* — умножение (*B*, *W*, *L*, *F*, *D*, *G*, *H*); 2 операнда.
30. *MUL_3* — умножение (*B*, *W*, *L*, *F*, *D*, *G*, *H*); 3 операнда.
31. *DIV_2* — деление (*B*, *W*, *L*, *F*, *D*, *G*, *H*); 2 операнда.
32. *DIV_3* — деление (*B*, *W*, *L*, *F*, *D*, *G*, *H*); 3 операнда.
33. *EMUL* — расширенное умножение.
34. *EDIV* — расширенное деление.
35. *EMOD_* — расширенное умножение (*F*, *D*, *G*, *H*) с выделением целой части.
36. *POLY_* — полином (*F*, *D*, *G*, *H*).
37. *INDEX* — индекс массива.

Инструкции работы с адресами

38. *MOVA_* — пересылка адреса (*B*, *W*, *L=F*, *Q=D=G*, *O=H*) — данных.
39. *PUSHA_* — загрузка в стек адреса (*B*, *W*, *L=F*, *D=Q=G*, *O=H*) — данных.

Инструкции работы с регистрами процессора

40. *PUSHR* — загрузка регистров в стек.
41. *POPR* — извлечение регистров из стека.
42. *MOVPSL* — пересылка расширенного слова состояния процессора.
43. *BI_PSW* — (*C*-сброс, *S*-установ) разрядов в слове состояния программы.

Инструкции переходов (безусловные и по кодам условий)

44. *BR_* — переход с (*B*, *W*) — смещением.
45. *JMP* — универсальный переход.
46. *BLSS_* — переход (знаковый, *U*-беззнаковый) по «меньше».
47. *BLEQ_* — переход (знаковый, *U*-беззнаковый) по «меньше или равно».
48. *BEQL_* — переход (знаковый, *U*-беззнаковый) по равенству.
49. *BNEQ_* — переход (знаковый, *U*-беззнаковый) по «неравенству».
50. *BGEQ_* — переход (знаковый, *U*-беззнаковый) по «больше или равно».
51. *BGTR_* — переход (знаковый, *U*-беззнаковый) по «больше».
52. *BV_* — переход по биту переполнения (*C*-сброшенному, *S*-установленному).
53. *BC_* — переход по биту переноса (*C*-сброшенному, *S*-установленному).

Инструкции сложных переходов

54. *ACB_* — сложение, сравнение (*B*, *W*, *L*, *F*, *D*, *G*, *H*) и переход.
55. *AOBLEQ* — прибавить единицу и перейти, если «меньше или равно».
56. *AOBLSS* — прибавить единицу и перейти, если «меньше».
57. *SOBGEQ* — вычесть единицу и перейти, если «больше или равно».
58. *SOBGTR* — вычесть единицу и перейти, если «больше».
59. *CASE_* — выбор (*B*, *W*, *L*).

Инструкции работы с подпрограммами и процедурами

60. *BSB_* — переход к подпрограмме с (*B*, *W*) — смещением.
61. *JSB* — универсальный переход к подпрограмме.
62. *RSB* — возврат из подпрограммы.
63. *CALLG* — вызов процедуры из общего вида.
64. *CALLS* — вызов из процедуры стекового вида.
65. *RET* — возврат из процедуры.

Инструкции для упакованных десятичных строк

66. *MOVP* — пересылка.
67. *CMPP3* — сравнение (3 операнда).
68. *CMPP4* — сравнение (4 операнда).
69. *ASHP* — арифметический сдвиг с округлением.
70. *ADDP4* — сложение (4 операнда).
71. *ADDP6* — сложение (6 операндов).
72. *SUBP4* — вычитание (4 операнда).
73. *SUBP6* — вычитание (6 операндов).
74. *MULP* — умножение.
75. *DIVP* — деление.
76. *CVTLP* — преобразование длинного слова в упакованную строку.
77. *CVTPL* — преобразование упакованной строки в длинное слово.
78. *CVTPT* — преобразование строки упакованной в числовую с завершающим знаком.
79. *CVTTP* — преобразование строки числовой с завершающим знаком в упакованную.
80. *CVTPS* — преобразование строки упакованной в числовую с лидирующим знаком.
81. *CVTSP* — преобразование строки числовой с лидирующим знаком в упакованную.
82. *EDITPC* — редактирование.

Инструкции для символьных строк

83. *MOVC3* — пересылка (3 операнда).
84. *MOVC5* — пересылка (5 операндов).
85. *MOVTC* — пересылка с перекодировкой.
86. *MOVTUC* — пересылка с перекодировкой до помеченного символа.
87. *CMPC3* — сравнение (3 операнда).
88. *CMPC5* — сравнение (5 операндов).
89. *LOCC* — локализация символа.
90. *SKPC* — пропуск символов.
91. *SPANC* — поиск соответствия.
92. *SCANC* — поиск несоответствия.
93. *MATCHC* — поиск подстроки.

Инструкции для битовых полей переменной длины

94. *EXTV* — чтение (извлечение) поля.
95. *EXTZV* — чтение поля с расширением лидирующими нулями.
96. *INSV* — запись (вставка) поля.
97. *CMPV* — сравнение поля.
98. *CMPZV* — сравнение поля с лидирующими нулями.
99. *FF_* — поиск первого (*C*-сброшенного, *S*-установленного) бита.
100. *BLB_* — переход, если младший бит (*C*-сброшен, *S*-установлен).
101. *BB_* — переход, если бит (*C* — сброшен, *S*-установлен).
102. *BB__* — переход, если бит (*C*-сброшен, *S*-установлен) и безусловно (*C*-сбросить, *S*-установить) бит.
103. *BBCCI* — переход, если бит сброшен, и безусловно сбросить бит (с блокировкой памяти).
104. *BBSSI* — переход, если бит установлен, и безусловно установить бит (с блокировкой памяти).

Инструкции для очередей

105. *INSQUE* — вставить в очередь.
106. *INSQHI* — вставить в начало очереди (с блокировкой памяти).
107. *INSQTI* — вставить в конец очереди (с блокировкой памяти).
108. *REMQUE* — удалить из очереди.
109. *REMQHI* — удалить из начала очереди (с блокировкой памяти).
110. *REMQTI* — удалить из конца очереди (с блокировкой памяти).

Инструкции специального назначения

111. *CHM_* — смена текущего режима на (*K*, *E*, *S*, *U*)-режим.
112. *PROBE_* — проверка допустимости (*R*-чтения, *W*-записи).
113. *SVPCTX* — сохранить контекст процесса.
114. *LDPCTX* — загрузить контекст процесса.
115. *MTPR* — запись в привилегированный регистр.
116. *MFPR* — чтение из привилегированного регистра.
117. *REI* — возврат из прерывания.
118. *CRC* — циклический контроль.
119. *BPT* — ловушка отладчика.
120. *XFC* — вызов специальной (микропрограммной) функции.
121. *NOP* — отсутствие операции.
122. *HALT* — останов.

Таким образом, согласно принятой классификации, имеется 122 типа инструкций, которые имеют 304 кода операции и 317 мнемоник и приведены в табл. 3 и 4. (Для удобства программирования некоторым одинаковым кодам операции присвоено несколько мнемоник).

Таблица 3

Однобайтные коды операции *ML*

M \ L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	HALT	NOP	REI	BPT	RET	RSB	LDPCTX	SVPCTX	CVTPS	CVTSP	INDEX	CRC	PROBER	PROBEW	INSQUE	REMQUE
1	BSBB	BRB	BNEQ	BEQL	BGTR	BLEQ	JSB	JMP	BGEQ	BLSS	BGTRU	BLEQU	BVC	BVS	BGEQU	BLSSU
1			BNEQV	BEQLU											BCC	BCS
2	ADDP4	ADDP6	SUBP4	SUBP6	CVTPT	MULP	CVTTP	DIVP	MOVC3	CMPC3	SCANC	SPANC	MOVC5	CMPC5	MOVTC	MOVTUC
3	BSBW	BRW	CVTWL	CVTWB	MOVP	CMPP3	CVTPL	CMPP4	EDITPC	MATCHC	LOCC	SKPC	MOVZWL	ACBW	MOVAW	PUSHAW
4	ADDF2	ADDF3	SUBF2	SUBF3	MULF2	MULF3	DIVF2	DIVF3	CVTFB	CVTFW	CVTFL	CVTRFL	CVTBF	CVTWF	CVTLF	ACBF
5	MOVF	CMPF	MNEGF	TSTF	EMODF	POLVF	CVTFD		ADAWI				INSQHI	INSQTI	REMQHI	REMQTI
6	ADDD2	ADDD3	SUBD2	SUBD3	MULD2	MULD3	DIVD2	DIVD3	CVTDB	CVTDW	CVTDL	CVTRDL	CVTBD	CVTWD	CVTLD	ACBD
7	MOVD	CMPD	MNEGD	TSTD	EMODD	POLVD	CVTDF		ASHL	ASHQ	EMUL	EDIV	CLRQ	MOVQ	MOVAQ	PUSHAQ
7													CLRD		MOVAD	PUSHAD
8	ADDB2	ADDB3	SUBB2	SUBB3	MULB2	MULB3	DIVB2	DIVB3	BISB2	BISB3	BICB2	BICB3	XORB2	XORB3	MNEGB	CASEB
9	MOVB	CMPB	MCOMB	BITB	CLRB	TSTB	INCB	DECB	CVTBL	CVTBW	MOVZBL	MOVZBW	ROTL	ACBB	MOVAB	PUSHAB
A	ADDW2	ADDW3	SUBW2	SUBW3	MULW2	MULW3	DIVW2	DIVW3	BISW2	BISBW3	BICW2	BICW3	XORWK	XORW3	MNEGW	CASEW
B	MOVM	CMPW	MCOMW	BITW	CLRW	TSTW	INCW	DECW	BISPSW	BICPSW	POPR	PUSHR	CHMR	CHME	CHMS	CHMU
C	ADDL2	ADDL3	SUBL2	SUBL3	MULL2	MULL3	DIVL2	DIVL3	BISL2	BISL3	BICL2	BICL3	XORL2	XORL3	MNEGL	CASEL
D	MOVL	CMPL	MCOML	BITL	CLRL	TSTL	INCL	DECL	ADWC	SBWC	MTPR	MFPR	MOVPSL	PUSHL	MOVAL	PUSHAL
D					CLPF										MOVAF	PUSHAF
E	BBS	BBC	BBSS	BBCS	BBSC	BBCC	BBSSI	BBCCI	BLBS	BLBC	FFS	FFC	CMRV	CMPZV	EXTV	EXTZV
F	INSV	ACBL	AOBLSS	AOBLEQ	SOBGEQ	SOBGTR	CVTLB	CVTLW	ASHP	CVTLP	CALLG	CALLS	XFC			

Примечание. *M* — младшая тетрада кода операции; *L* — старшая тетрада кода операции.

Таблица 4

Двухбайтные коды операции: *MLFD*

M \ L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																
2																
3			CVTDH	CVTGH												
4	ADDG2	ADDG3	SUBG2	SUBG3	MULG2	MULG3	DIVG2	DIVG3	CVTGB	CVTGW	CVTGL	CVTRGL	CVTBG	CVTWG	CVTLG	ACBG
5	MOVG	CMPG	MNEGG	TSTG	EMODG	POLVG	CVTGH									
6	ADDH2	ADDH3	SUBH2	SUBH3	MULH2	MULH3	DIVH2	DIVH3	CVTHB	CVTHW	CVTHL	CVTRHL	CVTBH	CVTWH	CVTLH	ACBH
7	MOVH	CMPH	MNEGH	TSTH	EMODH	POLVH	CVTHG						CLRH		MOVAH	PUSHAH
7													CLRO	MOVO	MOVAO	PUSHAO
8																
9									CVTFH	CVTFG						
A																
B																
C																
D																
E																
F							CVTHF	CVTHD								

Описание набора инструкций

Следующие разделы описывают функции большинства инструкций.

Инструкции для работы с целыми числами и числами с плавающей запятой

Логические и арифметические инструкции для целых чисел наглядно демонстрируют, каким образом коды операций, типы данных и режимы адресации могут сочетаться в инструкции. Большинство операций, предназначенных для целых чисел, используются также для чисел с плавающей запятой, а также для десятичных упакованных строк.

Арифметические инструкции включают как двухоперандную, так и трехоперандную формы. Двухоперандные инструкции записывают результат на место одного из двух операндов, например: $A=A+B$. Трехоперандные инструкции обеспечивают эффективное выполнение операторов языков высокого уровня, в которых две различные переменные используются для вычисления третьей, таких, как $A=B+C$. Инструкции с тремя операндами имеются для целочисленных данных, для данных в виде чисел с плавающей запятой и для данных в виде упакованных десятичных строк.

Для иллюстрации возможностей инструкций и режимов адресации рассмотрим оператор языка ФОРТРАН:

$$A(I)=B(I)-C(I),$$

где A, B и C являются статически расположенными массивами *REAL*∗4, а I — целое число *INTEGER*∗4. Последовательность инструкций, которая реализует этот оператор следующая:

```
MOVL I, R0
MULF3 B[R0], C[R0], A[R0].
```

Те же самые инструкции применяются, если A, B и C являются типами данных *REAL*∗8, *INTEGER*∗4, *INTEGER*∗2 или даже *INTEGER*∗1; в этом случае инструкция *MULF3* заменяется на *MULD3*, *MULL3*, *MULW3* или *MULB3* соответственно.

Если A, B и C динамически располагаемые массивы, то последовательность инструкции должна быть следующей:

```
MOVL I, R0
MULF3 B(FP) [R0], C(FP) [R0], A(FP) [R0].
```

Если A, B и C являются аргументами процедур, то имеем следующую последовательность:

```
MOVL I, R0
MULF3 @B(AP) [R0], @C(AP) [R0],
@A(AP) [R0]
```

Фактически расположение массивов A, B и C может выбираться произвольно. Например, комбинируя вышеуказанные способы таким образом, что A — статически расположен, B — динамически расположен, а C — аргумент, получаем следующую последовательность:

```
MOVL I, R0
MULF3 B(FP) [R0], @C(AP) [R0], A[R0]
```

Некоторые арифметические инструкции используются для увеличения точности повторяющихся вычислений:

инструкция «Расширенное умножение» (*EMUL*) умножает целочисленные двухсловные операнды и вырабатывает результат в виде учетверенного слова;

инструкция «Расширенное деление» (*EDIV*) делит учетверенное целочисленное слово на удвоенное слово и вырабатывает частное в виде удвоенного слова;

инструкция «Расширенное умножение с выделением целой части» (*EMOD*) умножает число с плавающей запятой на число с плавающей запятой расширенной точности (расширенной до 9 или 19 цифр точности для форматов *F* и *D* с 8 разрядами, отведенными на порядок) и выдает ответ отдельно в виде целой части и мантиссы. Такая инструкция особенно полезна для обработки аргумента тригонометрических и экспоненциальных функций через стандартный интервал;

инструкция «Полином» (*POLY*) вычисляет полином по таблице коэффициентов, используя метод Горнера. Эта инструкция используется в математической библиотеке языков высокого уровня для таких операций, как синус и косинус.

Инструкции для упакованных десятичных чисел

Многие из инструкций для данных в виде целых чисел и чисел с плавающей запятой имеют соответствующие модификации для упакованных десятичных строк:

«Пересылка» (*MOVP*) для пересылки упакованной десятичной строки из одной области в другую;

«Арифметический сдвиг с округлением» (*ASHP*) для перемещения десятичной точки на заданную величину с дополнительным округлением результата;

«Сравнение упакованное» (*CMPP*) для сравнения двух десятично-упакованных строк имеется два вида инструкции: трехоперандная для строк одинаковой длины (*CMPP3*) и четырехоперандная для строк различной длины (*CMPP4*);

инструкции преобразования (*CVT*___) осуществляют преобразования между упакованными десятичными строками и традиционно используемыми числовыми строками. Числовые строки могут быть с лидирующим или завершающим знаком и допускают различную кодировку: зонную или перфокарточную;

инструкции *ADDP* и *SUBP* предназначены для сложения или вычитания двух упакованных десятичных строк с возможностью либо замены слагаемого или вычитаемого на результат (*ADDP4* и *SUBP4*), либо загрузки результата в третью строку (*ADDP6* и *SUBP6*);

инструкции *MULP* и *DIVP* предназначены для умножения или деления двух упакованных десятичных строк и загрузки результата в третью строку;

инструкция «Редактирование» (*EDITPC*) преобразует данную десятично-упакованную строку в

строку символов, используя шаблон редактирования. Шаблон редактирования позволяет создавать числовые поля со следующими характеристиками: заполнение ведущими нулями, защитой ведущих нулей, защита от заполнения ведущими астериками (символ *), плавающий знак, плавающий валютный символ, представления специальных знаков, символы вставки, пробелы вместо незначащих нулей.

Инструкции для символьных строк

Эти инструкции производят операции над строками байтов:

инструкции «Пересылка» (*MOVC3* и *MOVC5*) просто копируют строки символов из одной области в другую. Они являются оптимальными при операциях пересылки блоков. 5-операндная модификация предусмотрена для работы с символом наполнителем (определяемым пользователем), который инструкция использует, чтобы дополнить область приемника до данного размера;

инструкции «Пересылка с перекодировкой» (*MOVTC*) и «Пересылка с перекодировкой до помеченного символа» (*MOVTUC*) в действительности формируют новые строки символов. Пользователь создает строку, которую инструкция использует как список в таблице перекодировки. Инструкция выбирает символы из таблицы в том порядке, в котором список смещения указывает на таблицу. Инструкция *MOVTC* позволяет пользователю создавать символ-наполнитель, который используется, чтобы дополнить строку-результат до заданного размера произвольным символом. Инструкция *MOVTUC* позволяет пользователю поддерживать любое количество символов переключения кода (*ESCAPE*). Когда следующее смещение указывает на знак переключения кода в таблице, перекодировка прекращается;

инструкция сравнения строк имеет 3-операндный (*CMPC3*) и 5-операндный (*CMPC5*) формат. Инструкции обоих форматов сравнивают две строки с начала до конца, информируя пользователя о нахождении первого несовпадающего символа или о достижении конца любой строки. 5-операндный тип предусмотрен для работы с символом-наполнителем, который используется инструкцией для эффективного заполнения строки при сравнении строк разной длины;

инструкции «Локализация символа» (*LOCC*) и «Пропуск символов» (*SKPC*) являются операциями поиска определенного символа внутри строки. *LOCC* просматривает данную строку до символа, который соответствует искомому символу, задаваемому пользователем. Это предпочтительно, например, при поиске ограничителя конца строки переменной длины. *SKPC*, с другой стороны, находит первый символ в строке, который отличается от заранее заданного в инструкции символа. Её рекомендуется использовать, когда надо пропустить наполняющие символы конца поля, чтобы найти начало следующего поля;

инструкция «Поиск подстроки» (*MATCHC*) похожа на инструкцию *LOCC*, но она определяет местонахождение не одиночного символа, а символьной строки. *MATCHC* просматривает строку до первого местонахождения подстроки, определенной пользователем;

инструкции «Поиск соответствия» (*SPANC*) и «Поиск несоответствия» (*SCANC*) являются операциями поиска, которые ищут членов заданного подмножества символов. Для этих инструкций пользователь задает строку символов, маску и адрес 256-байтной таблицы определения класса символа. Для каждого символа в данной строке инструкция ищет код класса в таблице, а затем выполняется конъюнкция с заданной маской и кодом класса символа. *SPANC* находит первый символ в строке, который соответствует классу, задаваемому маской. *SCANC* находит первый символ в строке, который относится к любому другому классу, кроме задаваемого его собственной маской.

Инструкции для работы с битовыми полями переменной длины

Инструкции обработки битовых полей переменной длины позволяют пользователю записывать, считывать и модифицировать эти поля, размер и адрес которых определяется исходя из беззнакового битового размера, базового адреса и знакового битового смещения. Если поле расположено в памяти, смещение может достигать битов, расположенных на расстоянии 2^{31} (приблизительно 256 млн байт) в любом направлении относительно базового адреса. Если поле расположено в регистре, смещение не может быть больше чем 31. Поля переменной длины (от 0 до 32 бит) могут использоваться для создания компактных заголовков, для кодов слова состояния или для создания пользовательских типов данных:

инструкции «Запись поля» (*INSV*) и «Чтение поля» (*EXTV* и *EXTZV*) загружают и извлекают данные в/из полей. Инструкция *INSV* берет определенное количество битов двойного слова (начиная с младшего бита) и записывает их в поле, которое может начинаться с любого бита, относительно заданного базового адреса. Инструкция «Чтение поля» перемещает данные из битового поля в младшие биты двойного слова. Поле может быть либо знаковым (*EXTV*), либо беззнаковым (*EXTZV*);

инструкции «Сравнение поля» (CMP) и «Поиск первого установленного/сброшенного бита» (*FF__*) применяются пользователем, чтобы проверить содержимое поля. Инструкция «Сравнения поля» извлекает поле и затем сравнивает его с данным словом двойной длины. Поле может интерпретироваться как знаковое (*CMPV*) и как беззнаковое (*CMPZV*). Инструкции поиска определяют первый бит в поле, который является нулевым (*FFC*) или установленным (*FFS*), просматривая поле от младших битов к старшим. Эти инструкции предпочтительны при анализе битовых структур. Например, двойное слово может представлять список очередей, обрабатываемых в порядке приоритета от 0 (высший) до 31 (низший). Каждый установленный бит представляет собой действующую очередь с высшим приоритетом среди действующих очередей. Вместе с инструкцией *SKPC* инструкции *FFC* и *FFS* оказываются полезными при

просмотре таблицы распределения произвольной длины.

Инструкции очередей

Имеется шесть инструкций, которые позволяют легко создавать и обслуживать структуры данных в виде очередей. Очереди, обрабатываемые с помощью этих инструкций представляют собой элементы данных в виде круговых двукратно-связанных списков. Первое двойное слово элемента очереди содержит указатель вперед к следующему элементу очереди, а следующее двойное слово содержит указатель назад на предшествующий элемент очереди.

Обеспечиваются два типа очередей: абсолютная и относительная. Абсолютные очереди используют указатели, которые являются виртуальными адресами, тогда как относительные очереди используют указатели, которые являются относительными смещениями.

Инструкции *INSQUE* и *REMQUE* предназначены для работы с абсолютными очередями. Инструкция *INSQUE* вставляет элемент, определяемый операндом-элементом в очередь, вслед за элементом, определяемым операндом-предшественником. *REMQUE* удаляет элемент, определяемый операндом-элементом. Элементы очереди могут располагаться на границе произвольного байта. Как *INSQUE*, так и *REMQUE* не могут быть прерваны до своего завершения.

Инструкции «Вставить элемент в начало очереди» (*INSQHI*) и «Удалить элемент из начала очереди» (*REMQHI*), «Вставить элемент в конец очереди» (*INSQTI*) и «Удалить элемент из конца очереди» (*REMQTI*) могут быть использованы при работе с относительными очередями. Кроме того, эти инструкции осуществляют блокировку памяти, что позволяет взаимодействующим процессам в многопроцессорных системах реализовывать доступ к разделяемым очередям без дополнительной синхронизации. Вход очереди должен представлять собой выравненное учетверенное слово.

Инструкции работы с адресами

Так как процессор предлагает множество режимов адресации, обеспечивающих легкий доступ к структурам данных через базовые адреса и индексы в регистрах, то адресами часто манипулируют. Имеется две инструкции:

«Переслать адрес» (*MOVA*) — это инструкция, которая пересылает адрес байта, слова, двойного слова (или числа с плавающей запятой одинарной точности), учетверенного слова (или числа с плавающей запятой двойной точности) в определенный регистр или ячейку памяти. Инструкция *MOVA* используется для загрузки базового регистра и выполнения вычислений позиционно-независимых адресов во время работы программы. Она имеет некоторые интересные применения, поскольку может использоваться как сложная операция сложения: *MOVAB X(R1)[R2], Y* — два сложения в одной команде; *MOVA__ X(Rn)[Rn], Rn* — умножение *Rn* на (*W, L, Q*) и сложение со смещением.

«Загрузить адрес в стек» (*PUSHA*) — эта инструкция, которая загружает адрес байта, слова, двойного слова (или числа с плавающей запятой одинарной точности), учетверенного слова (или числа с плавающей запятой двойной точности) в стек. Инструкция *PUSHA* используется при вычислении передаваемого адреса в вызываемых подпрограммах и процедурах.

Инструкции работы с регистрами процессора

Инструкция «Загрузка регистров в стек» (*PUSHR*) засылает несколько общих регистров в стек в течение одной операции. Пользователь задает слово-маску, в котором каждый установленный разряд (0—14) представляет соответствующий регистр (*R*0—*R*14), засылаемый в стек. Регистр *R*15 (счетчик программы) является единственным общим регистром, который не может быть загружен в стек этой инструкцией. Инструкция «Извлечение регистров стека» (*POPR*) является противоположной операцией, извлекающей каждый регистр следующими друг за другом словами двойной длины из стека, согласно заданному слову-маске. Инструкции *PUSHR* и *POPR* чрезвычайно эффективны при организации подпрограмм и обработки прерываний.

Инструкция «Пересылка расширенного слова состояния процессора» (*MOVPSL*) позволяет анализировать регистр *PSL*, загружая его содержимое в определенную ячейку.

Инструкции установа и сброса битов слова состояния программы (*BISPSW и BICPSW*) позволяют устанавливать или сбрасывать коды условий и биты особых ситуаций в регистре *PSW*. Маска определяет биты, которые следует установить или сбросить.

Инструкции переходов

Инструкции «Переход» и «Универсальный переход» являются основными инструкциями безусловной передачи управления. Обе инструкции (*BR* и *JMP*) загружают новые адреса в счетчик программы. Смещение (*B* или *W*), определяемое пользователем в инструкции *BR*, прибавляется к текущему содержимому счетчика программы, чтобы получить новый адрес. Инструкция универсального перехода (*JMP*) позволяет загружать адреса, определяемые пользователем, используя один из режимов адресации (кроме регистрового). Инструкции *JMP*, в отличие от *BR*, доступно все адресное пространство памяти.

Инструкции переходов по кодам условий позволяют только байтовые смещения и передавать управление, в зависимости от значения одного или более битов условия в слове состояния программы (*PSW*). Эти инструкции подразделяют на три группы:

знаковые переходы, которые используются для проверки результатов инструкций, оперирующих с целыми числами (или полями, обрабатываемыми как знаковые целые), числами с плавающей запятой и десятичными строками;

беззнаковые переходы, которые используются для проверки результатов инструкций, оперирующих над целыми числами (или полями, обрабатываемыми как беззнаковые целые), символьными строками и адресами;

инструкции, проверяющие переполнение и перенос, которые используются для проверки переполнения, когда не разрешены программные ловушки (для арифметики повышенной точности и для результатов специальных инструкций).

Мнемоники инструкций ясно показывают вариант проверки среди знаковых и беззнаковых интерпретаций типов данных в виде целых чисел. Проверки определяют, является ли результат предыдущей операции «меньше», «меньше или равен», «равен», «не равен», «больше или равен» или «больше». Например, инструкция «Беззнаковый переход по меньше или равно» (*BLEQU*) осуществляет переход, если установлен либо бит переноса (*C*), либо бит нулевого результата (*Z*). «Знаковый переход по больше» (*BGTR*) происходит, если ни бит отрицательного результата (*N*), ни бит нулевого результата (*Z*) не установлены.

Основное назначение инструкций перехода по значению одного бита то же, что и для инструкций условных переходов. Инструкции «Переход, если младший бит установлен» (*BLBS*) и «Переход, если младший бит сброшен» (*BLBC*) проверяют бит 0 операнда. Инструкции «Переход, если бит установлен» (*BBS*) и «Переход, если бит сброшен» (*BBC*) проверяют любой выбранный бит.

Имеются специальные типа инструкций перехода по значению бита, которые одновременно устанавливают или сбрасывают бит. Примером является инструкция «Переход, если бит установлен и безусловно установить бит» (*BBSS*): переход происходит, если заданный бит установлен, в противном случае выполняется следующая инструкция; в любом случае инструкция устанавливает заданный бит. Таким образом, инструкция *BBSS* может рассматриваться, как инструкция «Установка бита», с выполнением перехода, если бит уже был установлен.

Помимо *BBSS* существуют следующие инструкции: «Переход, если бит сброшен и безусловно сбросить бит» (*BBCC*); «Переход, если бит установлен и безусловно сбросить бит» (*BBSC*); «Переход, если бит сброшен и безусловно установить бит» (*BBCS*). Эти инструкции особенно предпочтительны для отслеживания инициализации или завершения процедуры и для сигнализации завершения или инициализации во взаимодействующих процессах (организация «семафоров»).

И, наконец, имеются две инструкции перехода по значению бита с блокировкой памяти, которые обеспечивают защиту анализируемого бита от внешнего воздействия: «Переход, если бит установлен и безусловно установить бит, с блокировкой» (*BBSSI*) и «Переход, если бит сброшен и безусловно сбросить бит, с блокировкой» (*BBCCI*).

Эти инструкции (*BBSSI* и *BBCCI*) обеспечивают блокировку памяти на время своего выполнения. Никакие другие операции не могут прервать эти инструкции, чтобы получить доступ к байту, содержащему управляющую переменную, в период между опросом бита и очисткой или установкой бита.

Инструкции для языков высокого уровня

Многие из описанных ранее инструкций эффективно реализуют некоторые конструкции языков программирования: инструкции трехадресной арифметики, расширенное умножение и деление и др. Помимо этого существует ряд других высокоуровневых инструкций.

Инструкция «Индекс массива» (*INDEX*) вычисляет индекс для массива элементов данных фиксированной длины (целых чисел, чисел с плавающей запятой, битовых полей, символьных строк и десятичных строк). Аргументами инструкции являются: нижний индекс, границы нижнего индекса, размер элемента массива, данный индекс и приемник для вычисленного индекса. Выполнение операции включает в себя контроль диапазона при вычислениях на языках высокого уровня, использующих границы нижнего индекса, и это позволяет оптимизировать вычисления индексной переменной массива.

Операторы языка КОБОЛ

```
01 A — ARRAY
 02 APICX (10) OCCURS 15 TIMES
 01 B PICX (10)
MOVE A (I) TO B
```

эквивалентны выполнению следующих инструкций

```
INDEX I, #1, #15, #10, #0, R0;
R0 <—— (0+1)*10
при 1=<I=<15
MOVC #10, A — 10[R0], B
```

Операторы языка Фортран

```
INTEGER*4A (L1:U1, L2:U2), I, J
A (I, J)=1
```

эквивалентны выполнению следующих инструкций

```
INDEX J, # L2, # U2, # M1, # 0, # 0, R0;
M1=U1 — L1
INDEX I, #L1, #U1, #1, R0, R0
MOVL #1, A — K[R0];
K=((L2*M1)+L1)*4
```

Имеется три типа инструкций перехода, которые могут быть использованы для эффективного написания циклов. При первом типе цикл обеспечивается вычитанием единицы и переходом. Счетчик переменной (задаваемый пользователем) уменьшается каждый раз при выполнении цикла. В инструкции «Вычесть единицу и перейти, если больше» (*SOBGTR*) цикл повторяется до тех пор, пока содержимое счетчика не станет равным 0. В инструкции «Вычесть единицу и перейти, если больше или равно» (*SOBGEQ*) цикл повторяется до появления отрицательного значения в счетчике.

Инструкция «Прибавить единицу и перейти, если меньше» (*AOBLSS*) относится ко второму типу инструкций организации циклов. Счетчик и предельное значение должны быть заданы пользователем. Счетчик увеличивается в конце цикла до предельного значения. В инструкции «Прибавить единицу и перейти, если меньше или равно» (*AOBLEQ*) цикл повторяется до тех пор, пока счетчик не превысит определенный пользователем предел.

Третий тип инструкций цикла, эффективно используемый для реализации оператора *DO* языка ФОРТРАН и оператора *FOR* языка БЕЙСИК, — это

инструкция «Сложение, сравнение и переход» (*ACB*). В этом случае пользователь должен определить предел, счетчик и величину шага. При каждой итерации инструкция прибавляет значение величины шага к счетчику и сравнивает содержимое его с предельным значением. Знак величины шага определяет логическое отношение сравнения: циклы со сравнением на «меньше или равно», если величина шага положительна, и со сравнением на «больше или равно», если величина шага отрицательна.

Процессор реализует также инструкцию «Выбор» (*CASE*) вычисляемого перехода, которая применяется в языках высокого уровня при реализации операторов *CASE* и *GO TO*. Чтобы выполнить эту инструкцию, пользователь должен задать таблицу смещений, содержащую различные адреса, индексируемые полученным значением, как селектором (переключателем). Перехода не происходит, если селектор оказывается за пределами таблицы.

Все перечисленные инструкции переходов допускают передачу управления в пределах 16-разрядного смещения.

Инструкции работы с подпрограммами и процедурами

Вызов подпрограмм обеспечивается двумя специальными инструкциями: «Переход к подпрограмме» (*BSB*) и «Универсальный переход к подпрограмме» (*JSB*). Как *BSB*, так и *JSB* сохраняют содержимое программного счетчика в стеке в качестве адреса возврата. В инструкции «Переход к подпрограмме» пользователь задает либо байт (*BSBB*), либо слово (*BSBW*) смещения. В инструкции «Универсальный переход к подпрограмме» применяется один из видов адресации.

Подпрограмма должна завершаться инструкцией «Возврат из подпрограмм» (*RSB*), которая извлекает первое двойное слово из стека и загружает его в счетчик программы.

Процедуры — это подпрограммы общего назначения, которые используют списки аргументов, автоматически передаваемые процессором. Инструкции вызова процедур позволяют языковым процессорам и операционной системе обеспечить стандартный интерфейс вызова: сохраняют только те регистры, которые используются процедурой; передают список аргументов процедуре;

обрабатывают указатели стека, кадра и аргумента; устанавливают биты разрешения арифметических ловушек.

Первое слово процедуры содержит маску входа, которая используется аналогично маске сохранения в инструкции «Загрузка регистров в стек». Каждый установленный бит маски (из 12 младших битов) представляет один из универсальных регистров (*R0—R11*), которые использует процедура. Инструкция вызова процедуры просматривает маску и сохраняет помеченные регистры в стеке, а также автоматически сохраняет содержимое регистров указателя кадра, указатели аргумента и счетчика программы.

Инструкция «Вызов процедуры общего вида» (*CALLG*) воспринимает в качестве операнда адрес списка аргументов и передает этот адрес в регистр указателя аргумента. Инструкция «Вызов процедуры стекового вида» (*CALLS*) передает список аргументов (помещаемых в стек пользователя), загружая регистр указателя аргумента адресом стека.

Для завершения выполнения процедуры используется инструкция «Возврат из процедуры» (*RET*). Она применяет регистр указателя кадра, чтобы найти регистры, подлежащие восстановлению, а также чтобы привести в порядок любые данные, хранящиеся в стеке, включая связи вложенных процедур.

Привилегированные инструкции

Эти инструкции подразделяются на пять типов, позволяющих программам текущего режима получать обслуживание от операционной системы без риска нарушить целостность всей системы: инструкции изменения режима (*CHMK*, *CHME*, *CHMS*, *CHMU*); инструкции проверки допустимости чтения и записи (*PROBER*, *PROBEW*); инструкции возврата из особых ситуаций и прерываний (*REI*); инструкции переключения контекста (*SVPCTX* и *LDPCTX*); инструкции обращения к внутренним регистрам процессора (*MFPR* и *MTPR*).

Программа пользовательского режима может получить привилегированное обслуживание, обращаясь к обслуживающим процедурам операционной системы с помощью инструкции типа «*CALL*». До того как процедура начнет действительно выполняться, диспетчер операционной системы выполняет соответствующую инструкцию смены режима. При смене режима происходит переход из одного режима только в тот же самый или в более привилегированный режим. Когда происходит переход из одного режима в другой, тип предыдущего режима помещается в поле предыдущего режима расширенного слова состояния процессора. Инструкция смены режима является фактически операцией передачи управления, которая может быть представлена как инструкция вызова обслуживающей программы операционной системы. Программа пользовательского режима может применять инструкции изменения режима, но так как при этом происходит передача управления операционной системе, непривилегированные пользователи не могут загружать программы для выполнения в любом более привилегированном режиме. Программы пользовательского режима могут применять инструкции изменения режима для обслуживания ловушек пользователя, а также для обеспечения универсальных обслуживаний в программах пользовательского режима.

Для процедур обслуживания, написанных чтобы выполняться в привилегированных режимах (ядро, исполнитель, супервизор), имеются инструкции, которые дают возможность проверять возможность доступа в память для чтения (*PROBER*) и записи (*PROBEW*) из вызывающей программы. Это позволяет операционной системе обеспечить выполнение программ обслуживания в более привилегированном режиме относительно вызывающих программ, предотвращая доступ в защищаемые области памяти.

Привилегированные процедуры обслуживания операционной системы и подпрограммы обслуживания прерываний и особых ситуаций используют инструкцию «Возврат из прерывания» (*REI*). Единственным способом перехода процессора в менее привилегированный режим является использование инструкции *REI*. Подобно инструкциям «Возврат из подпрограммы» и «Возврат из процедуры», *REI* восстанавливает счетчик программы (*PC*) и слово состояния программы (*PSW*), чтобы продолжить выполнение процесса с того места, где он был прерван. Однако *REI* выполняет и специальные функции. Например, *REI* проверяет, было ли поставлено в очередь какое-либо асинхронное прерывание для текущего процесса во время выполнения программ, обслуживающих прерывания и особые ситуации, и заботится о том, чтобы процессор воспринял их. Далее процессор следит за тем, чтобы режим, которому передается управление, был либо тем же самым, либо менее привилегированным, чем режим, в котором процессор находился при прерывании или особой ситуации. Таким образом, инструкция *REI* пригодна для всех программ, включая написанные пользователем программы обработки передач управления, но программа не может повысить свои привилегии, восстанавливая состояние процессора.

Когда операционная система планирует операцию переключения контекста, процедура переключения контекста использует инструкции «Сохранить контекст процесса» (*SVPCTX*) и «Загрузить контекст процесса» (*LDPCTX*), чтобы сохранить контекст текущего процесса и загрузить контекст другого процесса. Модифицируя содержимое одного из внутренних регистров процессора, процедура переключения контекста операционной системы определяет расположение контекста оборудования, которое должно быть загружено.

В состав внутренних регистров процессора входят регистры, которые определяют текущее выполнение процесса, регистры управления таймером и др. Инструкции «Запись в привилегированный регистр» (*MTPR*), «Чтение из привилегированного регистра» (*MFPR*) являются единственными инструкциями, имеющими доступ к внутренним регистрам процессора. Инструкции *MTPR* и *MFPR* могут применяться только в режиме ядра.

Инструкции специального назначения

Инструкция «Циклический контроль» (*CRC*) осуществляет полиномиальный циклический контроль по избыточности для данной строки длиной до 32 разрядов. Пользователь указывает строку, для которой должна быть выполнена данная инструкция, и таблицу функций полинома. Библиотека операционной системы включает в себя таблицы для стандартных функций, таких, как *CRC*-16.

Инструкция «Ловушка отладчика» (*BPT*) заставляет процессор выполнять обслуживание состояний режима ядра в соответствии с вектором особой ситуации. *BPT* пользуется утилитами операционной системы (например, отладчиком), но также может быть использована любым процессом, который предназначает для обслуживания условий инструкции *BPT*.

Инструкция «Вызов специальной функции» (*XFC*) выполняет созданные пользователем микропрограммы в перезаписываемой управляющей памяти процессора.

Инструкция «Отсутствие операции» (*NOP*) необходима при отладке автономных программ.

Инструкция «Останов» (*HALT*) является привилегированной командой, которая применяется только операционной системой для остановки процесса.

Режим совместимости

Под управлением операционной системы процессор может выполнять набор инструкций СМ-4/СМ 1420/СМ 1600 в контексте любого процесса. Работая в режиме совместимости, процессор интерпретирует инструкции, выполняемые в контексте текущего процесса ,как инструкции СМ-4/СМ 1420/СМ 1600.

Режим совместимости, в основном, дает возможность операционной системе обеспечить среду для выполнения большинства пользовательских программ, написанных для СМ-4/СМ 1420/СМ 1600, за исключением автономных программ.

В режиме совместимости присутствуют все регистры общего назначения и все режимы адресации СМ-4/СМ 1420/СМ 1600. Регистры режима совместимости от *R*0 до *R*6 представляют из себя 16 младших разрядов регистров *R*0—*R*6 собственного режима. Регистр *R*7 (счетчик программ режима совместимости) — это младшие биты регистра *R*15 (счетчика программы собственного режима). Регистры *R*8—*R*14 не задействованы в режиме совместимости. Следует заметить, что регистр *R*6 режима совместимости используется как указатель стека для временного хранения данных локальных программ, и что стеку локальных программ отводится место в программной области, а не в области управления.

Однако имеются некоторые ограничения на среду, которую операционная система может обеспечить программам СМ-4/СМ 1420/СМ 1600. Например, инструкции «Заслать в (из) предыдущего режима» (*MTPI/MFPI*) не могут быть смоделированы операционной системой, так как они не передают управление программам собственного режима.

Для режима совместимости эмулируется слово состояния процессора СМ-4/СМ 1420/СМ 1600. Только коды условий и бит трассы находятся в соответствии по отношению к СМ 1700. Все прерывания и особые ситуации, которые имеют место при работе процессора в режиме совместимости, вызывают переход процессора в собственный режим. Именно в этом режиме операционная система производит обслуживание прерываний и особых ситуаций.

Набор инструкций режима совместимости — это система инструкций СМ-4/СМ 1420/СМ 1600 со следующими исключениями:

привилегированные инструкции (*HALT*, *WAIT*, *RESET*, *SPL*, *MARK*) являются запрещенными;

инструкции ловушек (*BPT, IOT, EMT, TRAP*) заставляют процессор переходить в собственный режим, где либо ловушка должна быть обслужена, либо инструкция эмулируется;

инструкции «Засылка в (из) предыдущий режим инструкций (данных)» (*MFPI, MTPI, MFPD* и MTPD) выполняются точно так же, как если бы они выполнялись машиной СМ-4/СМ 1420/СМ 1600 в режиме пользователя. Они игнорируют уровень предыдущего режима доступа и выполняются как инструкции *PUSH* и *POP* по отношению к текущему стеку;

инструкции плавающей арифметики эмулируются программным обеспечением.

Все остальные инструкции выполняются так, как если бы они выполнялись процессором СМ-4/СМ 1420/СМ 1600 в пользовательском режиме.

ФУНКЦИОНАЛЬНОЕ ОПИСАНИЕ АППАРАТНЫХ СРЕДСТВ

ВК СМ 1700 проектировался исходя из требований низкой стоимости и с учетом эффективной поддержки широких возможностей, предоставляемых системным программным обеспечением (ПО). Это привело к высокой алгоритмической сложности аппаратных средств, что в свою очередь привело к введению в ВК СМ 1700 дополнительных аппаратно-микропрограммных средств для обеспечения эффективной диагностики в процессе наладки и эксплуатации.

Консольно-диагностический процессор (КДП)

КДП является основным средством связи между оператором и ЭВМ СМ 1700. Он выполняет действия, задаваемые с помощью переключателей на передней панели управления или с консольного терминала, осуществляет контроль параметров сетевого напряжения и вторичных источников питания, расположенных в стойке вычислительной машины.

КДП обеспечивает загрузку и хранение микропрограмм ЦП, загрузку операционной системы, а также используется при диагностике и наладке машины.

В состав КДП входят: 8-разрядный микропроцессор; оперативная память микропроцессора; интервальный таймер.

Консольный терминал СМ6380 подключается к КДП через стык С2-ИС. Скорость связи с терминалом определяется переключателем и лежит в диапазоне 300...9600 бод. Через стык С2-ИС к КДП подключается консольный загрузчик СМ5218, представляющий собой сдвоенный накопитель на кассетной магнитной ленте. Скорость передачи 38,4 кбод. Кроме того, через стык С2 к КДП возможно подключение модемной линии, которая может применяться для работы с удаленным консольным терминалом в центре обслуживания.

Арифметико-логический процессор (АЛП)

АЛП предназначен для выполнения арифметических и логических операций, реализующих набор инструкций СМ 1700.

Восемь 4-разрядных микропроцессорных секций К1804ВС1 образуют 32-разрядное арифметико-логическое устройство процессора. В процессоре имеется местная память объемом 256 32-разрядных слов, которая используется для хранения и обработки содержимого программно-доступных регистров общего назначения и привилегированных регистров, а также в качестве рабочей памяти данных микропрограмм.

В процессоре также имеется 4-байтный регистр предвыборки, позволяющий выборку следующей команды совместить с выполнением текущей. Аппаратно поддерживается обработка следующих типов данных: байт, слово (16 разрядов), двойное слово (32 разряда). Длительность цикла процессора 270 нс.

Управление процессором осуществляется с помощью микропрограмм, которые хранятся в перезаписываемой памяти микрокоманд объемом 16К 24-разрядных слов. Дополнительные 4К слов могут быть предоставлены для управления интегрированным контроллером диска и микропрограмм пользователей. Микропрограммы записываются в память микрокоманд консольно запоминающим процессором при инициализации системы.

Устройство микропрограммного управления содержит микропрограммный стек, позволяющий эффективно использовать микроподпрограммы. Используется восемь форматов микрокоманд. Язык микропрограммирования МИАСС СМ позволяет создавать макроконструкции для качественного описания алгоритмов микропрограмм.

Контроллер оперативной памяти (КОП)

КОП предназначен для обеспечения обмена информацией между ОЗУ и остальными компонентами комплекса.

Контроллер выполняет управление ОЗУ емкостью до 5 Мбайт, поддерживает страничную организацию памяти, обеспечивает работу процессора без блокировки при обращении к памяти по каналу прямого доступа, выполняет трансляцию виртуальных адресов (32-разрядных, поступающих от ЦП или ИКД, и 18-разрядных, поступающих от устройств интерфейса ОШ) в физические 24-разряд-

ные адреса, осуществляет работу канала прямого доступа согласно требованиям интерфейса ОШ, выполняет защиту от несанкционированных обращений к ОЗУ.

Управление функциями контроллера реализовано на основе микропрограммного автомата. Цикл выполнения микрокоманд контроллера — 90 нс; разрядность микрокоманд — 72 бит; объем постоянного ЗУ микрокоманд — 512 слов. Контроллер способен осуществлять обращения к ОЗУ как по виртуальным, так и непосредственно по физическим адресам. При обменах между ОЗУ и процессором размерность пересылаемой информации — 32 бит, а при обменах с устройствами интерфейса ОШ — 16 бит. В информации, считанной из ОЗУ, контроллер обеспечивает коррекцию одиночных ошибок и обнаружение двойных ошибок.

Контроллер ОЗУ обеспечивает выборку информации из модуля ОЗУ через 700 нс, цикл чтения и записи — 810 нс.

Модуль ОЗУ

ОЗУ применяется в качестве накопителя наращиваемой оперативной памяти.

Модуль ОЗУ имеет емкость 1 Мбайт. Максимальный объем оперативной памяти СМ 1700 равен 5 Мбайт. При этом устанавливается пять модулей ОЗУ.

Обмен данными с модулем ОЗУ происходит словами по 39 разрядов, из которых 32-информационные и 7 — контрольные. Контрольные разряды обеспечивают обнаружение и исправление всех одиночных ошибок при чтении из модуля ОЗУ. Двойные ошибки обнаруживаются, но не исправляются. Схема обнаружения и коррекции ошибок является частью логики контроллера памяти.

Модуль ОЗУ содержит 156 микросхем К565РУ5Г, разделенных на четыре банка по 39 в каждом. Каждая микросхема содержит матрицу размером 256×256, реализующую 64К однобитовых ячеек. Следовательно, в каждом банке реализуется 64К 39-разрядных данных.

Операциями обращения к модулю ОЗУ являются чтение, запись и регенерация информации. Время выборки информации 700 нс. Цикл обращения к модулю ОЗУ не превышает 810 нс. Цикл регенерации требует 720 нс и выполняется через 12,8 мкс для каждой строки микросхемы, т. е. регенерация для всех микросхем памяти выполняется за 3,4 мс.

Процессор с плавающей запятой (ППЗ)

ВК СМ 1700 имеет набор инструкций для обработки чисел с плавающей запятой. Этот набор в СМ 1700 может быть реализован или чисто микропрограммным способом на аппаратуре центрального процессора СМ 1700, или на специальной аппаратуре со своим микропрограммным управлением, называемой процессором арифметики с плавающей запятой или просто ППЗ. Последняя реализация по сравнению с первой дает выигрыш в быстродействии в 5—7 раз в зависимости от обрабатываемых форматов чисел. Поэтому все серийные образцы СМ 1700 обеспечиваются аппаратным модулем ППЗ.

ППЗ реализует 98 инструкций и оперирует с четырьмя форматами данных — от 32 до 128 разрядов. ППЗ представляет собой расширение ЦП и самостоятельно работать не может. Исходные операнды ППЗ получает от ЦП партиями по 32 разряда, используя двунаправленную шину данных. По этой же шине результат пересылается из ППЗ в ЦП и далее в оперативную память. Основными функциональными единицами ППЗ являются: схема путей данных, дешифратор команд и блок микропрограммного управления.

ППЗ использует горизонтальную структуру микропрограммного управления с 48-разрядным форматом микрокоманды. Объем управляющей памяти — 1К микрокоманд. Длительность цикла микрокоманды ППЗ — 180 нс, т. е. составляет 2/3 длительности цикла ЦП. Блок микропрограммного управления синхронизирует свою работу с устройством управления ЦП и запускается в работу после получения кода операции из ЦП. После его дешифрации ППЗ начинает работу под управлением своей микропрограммы. ЦП в это время находится в режиме ожидания или вычисляет адрес памяти приемника, по которому будет записан результат (для 3-адресных инструкций). Получив результат, ППЗ пересылает его в ЦП и переходит в состояние ожидания следующей инструкции.

Контроллеры внешней памяти

В СМ 1700 предусмотрен ряд контроллеров внешней памяти, обеспечивающих подключение широкого спектра накопителей на магнитных дисках и магнитных лентах.

Интегрированный контроллер дисков (ИКД) СМ 1700.5129

Схемотехнически интегрированный контроллер дисков (СМ 1700.5129) выполнен как микропрограммный автомат, интегрированный с аппаратурой ЦП. Время цикла выполнения микрокоманды переменное — от 270 до 155 нс в зависимости от выполняемой контроллером операции. Длина слова микрокоманды — 64 бит, количество микрокоманд — 512.

Контроллер предназначен для подключения к внутренней шине процессора СМ 1700 до двух накопителей на магнитных дисках СМ 5504 (типа «Винчестер»).

ТЕХНИЧЕСКИЕ ДАННЫЕ

Диаметр носителя, мм	356
Способ записи	МЧМ
Емкость, Мбайт:	
неформатированная	160
форматированная	121
Количество:	
рабочих поверхностей	7
сервоповерхностей	1
рабочих головок	14
сервоголовок	1
Скорость передачи данных, Мбит/с	6,45

Подключение накопителей к контроллеру осуществляется через интерфейс СМД. Обмен данными между контроллером и оперативной памятью производится под управлением процессора. Контроллер обеспечивает буферизацию передаваемых данных. Он содержит два буфера по 512 байт каждый.

Для повышения достоверности считанной информации контроллер каждый записанный на диск сектор сопровождает 32-разрядным контрольным словом, что позволяет корректировать любое искажение информации в пачке длиной 11 бит (используется циклический корректирующий код полиномиального вида *CRC/ECC*). Контроллер реализует также механизм обхода дефектного сектора на дорожке, обеспечивая автоматическую переадресацию секторов.

Контроллер содержит восемь программно-доступных регистров, три из которых расположены в аппаратуре процессора СМ 1700 (все регистры имеют адреса на ОШ). Контроллер выполняет восемь программных и одну диагностическую команду, с помощью которой аппаратура контроллера проверяется в диагностическом режиме.

Непосредственное управление контроллера организуется собственным микропрограммным управлением, а управление выполнением программных команд ведется процессором ВК СМ 1700.

Контроллер магнитных дисков (КМД) СМ 5134

Предназначен для подключения к интерфейсу ОШ накопителей на магнитных дисках емкостью до 300 Мбайт, выходящих на интерфейс СМД (например, СМ 5504 емкостью 160 Мбайт или СМ 5511 емкостью 300 Мбайт).

ТЕХНИЧЕСКИЕ ДАННЫЕ

Максимальное количество подключаемых накопителей	4
Количество адресуемых регистров	20
Базовый адрес регистров контроллера	(776700)
Вектор прерывания	254
Уровень приоритета прерывания	5

Контроллер обеспечивает выполнение 19 операций, которые делятся на четыре группы:

операции управления состоянием (НЕТ ОПЕРАЦИИ, СБРОС НАКОПИТЕЛЯ, ОСВОБОДИТЬ НАКОПИТЕЛЬ, НАЧАЛЬНАЯ УСТАНОВКА ПЕРЕД ЧТЕНИЕМ, ПОДТВЕРДИТЬ ПАКЕТ);

операции позицирования (ПОИСК, ВОССТАНОВЛЕНИЕ, СМЕЩЕНИЕ, ВОЗВРАТ К ЦЕНТРУ ДОРОЖКИ, ПОИСК СЕКТОРА);

операции передачи данных (ЗАПИСЬ ДАННЫХ, ЗАПИСЬ ЗАГОЛОВКА И ДАННЫХ, ЧТЕНИЕ ДАННЫХ, ЧТЕНИЕ ЗАГОЛОВКА И ДАННЫХ, КОНТРОЛЬ ЗАПИСИ ДАННЫХ, КОНТРОЛЬ ЗАПИСИ ЗАГОЛОВКА И ДАННЫХ);

дополнительные операции (ЗАГРУЗКА, ФОРМАТ, УСТАНОВКА ШИРИНЫ ОБМЕНА).

Скорость передачи информации — 0,8 Мбайт/с. Обмен данными между контроллером и оперативной памятью производится по уровню «Прямой доступ». Для согласования пропускной способности канала ОШ со скоростью обмена данными с диском контроллер обеспечивает буферизацию трех секторов данных (по 512 байт каждый).

Ядром контроллера является 16-разрядный процессор, построенный на микропроцессорном наборе К1804.

Управление дисковым интерфейсом осуществляется полностью микропрограммно, что позволяет легко перестраивать контроллер для управления накопителями разного типа.

Емкость памяти микрокоманд — 512 слов, длина слова микрокоманды — 48 бит, время цикла выполнения микрокоманды — 150 нс.

При включении питания в контроллере запускается тест самодиагностики, обеспечивающий проверку не менее 70% аппаратуры. В случае ошибки загорается индикатор «Неисправность» и обращение к контроллеру блокируется.

Контроллер магнитных дисков (КМД) СМ 5130

Предназначен для подключения накопителей на магнитных дисках СМ 5514 (типа «Винчестер» диаметром 130 мм) емкостью 20—28 Мбайт (форматированная емкость 17,7—23,6 Мбайт) к интерфейсу ОШ ВК СМ 1700.

ТЕХНИЧЕСКИЕ ДАННЫЕ

Количество подключаемых накопителей	до 4
Количество одновременно обслуживаемых накопителей	1
Номинальная скорость обмена с накопителем, Мбит/с	5
Способ записи на диске	МЧМ
Форматированная емкость сектора, байт	512
Число цилиндров	320
Число дорожек на цилиндре	6 или 8
Интерфейс связи с накопителем	ИМДМ (*ST506*)
Среднее время позиционирования, мс	40
Обмен данными с ЭВМ	по уровню прямого доступа
Приоритет прерывания	по уровню 5

Контроль и исправление ошибок обмена данными с накопителем осуществляется по 32-разрядному циклическому коду, позволяющему исправление пакета одиночных ошибок до 11 бит.

Контроллер СМ 5130 выполняет восемь команд: выбор накопителя, возврат, поиск, чтение адреса (заголовка), запись адресов (форматирование), чтение и запись данных, проверка записи.

Контроллер СМ 5130 является архитектурным аналогом контроллера СМ 5408.5112 (используется в СМ 1600) и содержит 16 программно-доступных регистров.

Для согласования пропускной способности канала ОШ со скоростью обмена с накопителем в контроллере имеется буферная память на два сектора данных.

Ядром схемы контроллера является 16-разрядный микропроцессор серии К1804. (Время цикла

выполнения микрокоманды 200 нс. Длина слова микрокоманды 56 бит. Количество микрокоманд 512).

При включении питания в контроллере выполняется внутренний тест микродиагностики, обеспечивающий проверку основной аппаратуры контроллера. Во время выполнения теста микродиагностики на плате контроллера горит светоизлучающий диод, который после благополучного завершения диагностики гаснет.

Контроллер магнитных дисков (КМД) СМ 5408.5112

Данный контроллер — единственное изделие, которое не разрабатывалось специально для СМ 1700, а было заимствовано из ВК СМ 1600. Этот контроллер представляет из себя автономный комплектный блок, встраиваемый в отдельную стойку вместе с накопителями СМ 5408, имеющими сменные носители, каждый емкостью 16 Мбайт (форматированная емкость 14 Мбайт). Количество подключаемых накопителей до 8.

Контроллер СМ 5408.5112 подключается к ОШ через расширитель интерфейса, что позволяет скомпенсировать отрицательный эффект от удлинения и рассогласования ОШ и тем самым достичь удовлетворительных показателей помехоустойчивости.

Контроллер магнитных лент (КМЛ) СМ 5015

Предназначен для подключения к интерфейсу ОШ накопителей на магнитной ленте, имеющих индустриальный интерфейс (ИНМЛ-П).

Для НМЛ СМ 5309 индустриальный интерфейс обеспечивается форматером. Через один форматер к контроллеру СМ 5015 можно подсоединить до четырех накопителей СМ 5309.

ТЕХНИЧЕСКИЕ ДАННЫЕ

Максимальная емкость, Мбайт	40
Способ записи	БВН-1 или ФК
Плотность записи, бит/мм	32 или 63
Скорость обмена информацией, Кбайт/с	36 или 72
Рабочая скорость движения ленты, м/с	1,14
Формат записи в соответствии с требованиями	*TS*1863 и *IS*03788
Максимальный диаметр используемой бобины, мм	267

Схемотехнически контроллер СМ 5015 выполнен как 8-разрядный процессор, построенный на микропроцессорном наборе КМ1804. (Длина слова микрокоманды 56 бит. Количество микрокоманд 512. Время цикла выполнения микрокоманды 167 нс.).

Контроллер выполняет буферизацию передаваемых данных: он содержит буфер на 1024 байт.

При включении питания в контроллере запускается тест самодиагностики, обеспечивающий проверку исправности аппаратуры. В случае исправности загорается индикатор «Готовность».

Многофункциональный контроллер связи (МКС)

МКС подключается к системному интерфейсу ОШ и обладает возможностью прямого доступа к ОЗУ. Функционально контроллер связи состоит из отдельных устройств: асинхронного мультиплексора, синхронного адаптера дистанционной связи и параллельного 16-разрядного порта. В состав ВК СМ 1700 может входить до двух МКС.

Асинхронный мультиплексор обслуживает 8 линий, каждая из которых может быть запрограммирована на одну из 16 возможных скоростей в диапазоне 50...19 200 бод. Уровни сигналов асинхронного мультиплексора соответствуют стыку С2-ИС, что позволяет подключать на расстоянии до 500 м широкую номенклатуру алфавитно-цифровых видеотерминалов (АЦВТ) и устройств графического ввода-вывода (УГВВ).

Первые две линии имеют полное модемное управление по стыку С2-ИС и могут использоваться для подключения удаленных терминалов. К остальным шести линиям могут подключаться либо непосредственно устройства, имеющие выход на интерфейс С2-ИС, либо через преобразователь устройства, имеющие выход на интерфейс ИРПС. Преобразователь интерфейсов С2-ИС/ИРПС представляет собой дополнительный наконечник к кабелю устройства.

Асинхронный мультиплексор обеспечивает работу в дуплексном или полудуплексном режиме. Формат передаваемых данных — асинхронный последовательный с одним стартовым битом и одним или двумя стоповыми битами. Размер символа управляется программно (5, 6, 7 или 8 бит); паритет также управляется программно.

Принятые символы вместе с соответствующим номером линии заносятся в приемный буфер объемом 48 слов, общий для всех линий. Передача символов может производиться с использованием буферов, когда каждая линия передает символы из своего собственного 32-символьного буфера, или в режиме прямого доступа к ОЗУ.

Синхронный адаптер является одноканальным устройством, предназначенным для организации межмашинной связи, и осуществляет прием и передачу символов из ОЗУ по прямому доступу с двойной буферизацией. Для передачи и приема имеется два набора регистров адреса и счетчиков байтов. Поэтому передача и прием нового сообщения могут начинаться до того, как процессор оповещен о завершении предыдущего сообщения.

Синхронный канал можно запрограммировать на одну из восьми скоростей 800...19200 бод, которые задаются внутренним генератором. При внешней синхронизации (нормальный режим) скорость определяется тактовой частотой от модема.

Синхронный адаптер поддерживает биториентированные протоколы типа *HDLC* и байториентированный протокол типа *DDCMP*, который принят в архитектуре однородных сетей СМ ЭВМ. Для биториентированных протоколов адаптер выполняет широкий набор функций, включая обрамление кадров, битстаффинг и битстриппинг, подсчет контрольной суммы при приеме и передаче и распоз-

нование управляющих битовых последовательностей. Для байториентированного протокола *DDCMP* адаптер выполняет обрамление кадров, подсчет контрольной суммы, обработку управляющих символов. В адаптере может быть реализован любой байториентированный протокол. В этом случае адаптер выполняет только прямую пересылку данных между ОЗУ и синхронным интерфейсом. Функции, специфические для протоколов, должны отрабатываться сетевым программным обеспечением.

Синхронный адаптер включает полное модемное управление, уровни сигналов соответствует стыку С2-ИС.

Параллельный 16-разрядный порт может использоваться в одном из двух режимов: в режиме контроллера построчной печати с интерфейсом ИРПР или в режиме дуплексного/полудуплексного регистра связи с прямым доступом. При сборке вычислительного комплекса пользователь выбирает один из режимов с помощью переключателя.

Контроллер печати использует прямой доступ к ОЗУ и выполняет следующие функции форматирования на нижнем уровне: расширенную табуляцию, автоматическую вставку символа «Возврата каретки», автоматический перенос строки, преобразование смещения формата в несколько смещений строк.

Регистр связи может использоваться для различных целей, в том числе и для организации быстрого обмена между двумя СМ 1700 (удаление не более 15 м).

Контроллер локальной сети (КЛС)

Предназначен для обеспечения работы СМ 1700 в составе локальной сети магистрального типа, удовлетворяющей протоколу ISO8802/3 («Эзернет»). Скорость обмена между узлами сети до 10 Мбит/с, расстояние между узлами до 500 м.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Архитектурная поддержка программного обеспечения

Программное обеспечение (ПО) ВК СМ 1700 включает в себя: операционные системы; программное обеспечение для организации и управления базами данных; программные средства распределенной обработки данных; программные средства машинной графики и САПР; систему программного диагностирования ВК СМ 1700.

Эффективность построения ПО существенным образом определяется аппаратными средствами, заложенными в архитектуру ВК СМ 1700, направленными на достижение эффективности и рациональности реализации основных функций ПО в части управления процессором, памятью и вводом-выводом.

Необходимо выделить следующие основные особенности архитектуры ВК СМ 1700:

наборы инструкций СМ 1700 позволяют очень эффективно реализовывать основные программные процедуры операционной системы. К таким инструкциям в первую очередь относятся наборы инструкций для работы с битовыми полями переменной длины (сравнение, извлечение, вставка полей, поиск битов в поле, условные переходы) и очередями (занесение и удаление элементов очереди). Особенно следует отметить наличие инструкций, выполнение которых осуществляется с блокировкой памяти от параллельного доступа, что позволяет реализовывать «семафоры» для синхронизации работы нескольких процессоров с общей памятью;

процессор СМ 1700 может работать в четырех режимах, используя в каждом из них индивидуальные указатели стека. Это позволяет реализовывать многоуровневую структуру операционной системы;

наличие аппаратного контекста с возможностью его переключения позволяет использовать в операционной системе новую единицу управления-процесс;

аппаратура трансляции адресов процессора и ОШ позволяют операционной системе предоставлять программам пользователя практически неограниченный объем виртуальной памяти;

возможность префиксации области векторов прерывания позволяет управлять многопроцессорными вычислительными комплексами на базе СМ 1700;

наличие 15 уровней программно-генерируемых прерываний позволяет упростить механизм взаимодействия и синхронизации вспомогательных процессов в операционной системе;

в СМ 1700 реализован механизм аппаратной генерации программного прерывания второго уровня в момент перехода процессора в указанный пользователем режим (*ASTLVL*). Этот механизм позволяет откладывать прерывание, предназначенное для низкоприоритетного процесса.

Ниже более подробно рассматриваются вопросы использования в ПО перечисленных особенностей архитектуры СМ 1700.

Единицей планируемой работы в операционной системе СМ 1700 является процесс. Каждый процесс определяется совокупностью контекстов аппаратных средств и ПО и описанием адресного пространства. Каждая программа в операционной

системе выполняется в контексте процесса и называется образом. Процесс можно сопоставить с виртуальной машиной, которая определяет адресное пространство и логические ресурсы, распределенные пользователю. При этом виртуальная машина отождествляется с концептуальной средой, с которой взаимодействует программа, эта среда может совпадать или не совпадать с физическими ресурсами. Информация о каждом процессе хранится в нескольких структурных данных, расположенных в различных областях адресного пространства, распределенного для процесса.

Структурно процесс в системе состоит из контекста ПО, блока управления процессом, стеков процесса и адресного пространства процесса. Контекст ПО содержит управляющую информацию о процессе, используемую операционной системой для планирования и управления процессом при его выполнении: приоритет процесса; привилегии и ресурсы процесса; идентификатор процесса и др.

Блок управления процессом содержит контекст аппаратных средств процесса: значения универсальных регистров процессора; указатели стеков процесса; значения регистров процессора, используемых для управления памятью; слово состояния процессора; значения регистров, используемых для обработки асинхронных системных прерываний.

Процессор СМ 1700 выполняет инструкции в одном из четырех режимов: ядра, управления, супервизора и пользователя. В соответствии с этим операционная система обеспечивает обслуживание процессов в одном из четырех режимов, а процессор СМ 1700 обеспечивает механизм защиты программ, основанный на четырех режимах доступа к процессору. Каждый режим процесса определяет его привилегии в части доступа к памяти и возможность использования процессом различных типов команд.

Таким образом обеспечивается выделение в операционной системе четырех слоев, защищаемых аппаратно от доступа менее привилегированных слоев к более привилегированным. Программные компоненты операционной системы, выполняемые в одном из режимов, обеспечиваются защитой от непосредственного доступа программ, выполняемых в менее привилегированном режиме. Таким же образом осуществляется ограничение для доступа к некоторым структурам данных операционной системы, доступ к которым разрешается только из наиболее привилегированных режимов.

Каждый выполняемый процесс имеет четыре стека, используемых в зависимости от текущего режима доступа к процессору. Каждый стек защищается от доступа к нему из менее привилегированного режима. Введение отдельных стеков для каждого режима позволяет выполнять сохранение информации в текущем стеке без необходимости его защиты при изменении режима процесса на менее привилегированный.

Кроме четырех стеков, используемых процессом в одном из четырех режимов доступа к процессору, в СМ 1700 существует стек, используемый системой для обслуживания прерываний — стек прерываний.

Часть информации о состоянии процессора хранится в аппаратном регистре, называемым длинным словом состояния процессора (*PSL*).

Слоистая структура операционной системы, как уже указывалось, предполагает возможность выполнения различных функций системы в различных режимах процессора. Если программа, выполняющая некоторую функцию, должна выполняться в более привилегированном режиме, требуется изменение режима процессора. Например, запрос из пользовательской программы на ввод-вывод непосредственно реализуется в режиме ядра. Переключение режимов процессора осуществляется с помощью специальных инструкций изменения режимов доступа — *CHMK*, *CHME*, *CHMS*, обеспечивающих соответственно переходы в режимы ядра, выполнения, супервизора и пользователя. Эти инструкции выполняют переход из менее привилегированного режима к более привилегированному и имеют один операнд, который является кодом, определяющим привилегированную функцию или операцию, которую требуется выполнить.

Содержимое *PSL* и *PC* для предыдущего режима процесса сохраняется в текущем (новом) стеке. Код предыдущего режима процессора сохраняется в поле предыдущего режима, а код нового режима — в поле текущего режима *PSL*. Таким образом обеспечивается возможность в более привилегированном режиме определить предыдущий режим. Программа, получившая управление, в новом режиме доступа к процессору анализирует передаваемый в инструкции изменения режима код операнда и передает управление соответствующей компоненте операционной системы для выполнения запрошенной функции.

Системная программа, связанная с выполнением требуемой функции, выполняется при этом в контексте процесса пользователя. Таким образом, хотя системные программы выполняются в более привилегированном режиме (большинство в режиме ядра), они имеют полный доступ ко всему адресному пространству пользовательского процесса. В этом смысле эти программы не отличаются от обычных подпрограмм, а инструкции изменения режима доступа к процессору можно рассматривать как обращение к программам, которые осуществляются через границы между режимами доступа. Так как все инструкции изменения режимов контролируются специальной системной программой (диспетчером), исключается возможность некорректного использования инструкций изменения режимов для перехода в более привилегированный режим.

При завершении выполнения системных программ единственным способом перехода к менее привилегированному режиму (возврату управления к пользовательской программе) является выполнение инструкции выхода из прерывания *REI*. При выполнении инструкции *REI*, в вершине текущего стека должны находиться *PSL* и *PC* для нового (менее привилегированного) режима. Путем анализа поля текущего режима *PSL*, находящегося в стеке, проверяется, является ли новый режим менее привилегированным по сравнению с текущим. Таким образом обеспечивается недоступность перехода по инструкции *REI* к более привилегированному. Извлекаемые из стека по инструкции *REI*, *PSL* и *PC* определяют после выполнения инструкции новое состояние процессора и соответственно переключение на новый указатель стека, а также адрес выполнения программы в новом режиме.

Аппаратный контекст процесса в системе определяется блоком управления процессом (*PCB*). *PCB* представляет собой структуру данных, в которой сохраняются образы 14 универсальных регистров (*R*0 — *R*13), четырех указателей стека для каждого режима доступа к процессору (*KSP*, *ESP*, *SSP*, *USP*), двойного слова состояния процессора (*PSL*), счетчика инструкций (*PC*), регистров, используемых для управления памятью процессора (*POBR*, *POLR*, *PIBR*, *PILR*) и другие управляющие поля. При переключении контекста процессов содержимое аппаратных регистров для текущего процесса сохраняется в *PCB* этого процесса, а аппаратные регистры загружаются значениями (образами) этих регистров из *PCB* нового процесса. Адрес *PCB* текущего процесса всегда хранится во внутреннем привилегированном регистре СМ 1700, называемом регистром базы блока управления процессом (*PCBB*). Процедура сохранения состояния текущего процесса осуществляется с помощью инструкции *SVPCTX*, которая аппаратно реализует сохранение всех регистров в блоке *PCB*, адрес которого содержится в регистре *PCBB*. Затем в регистр *PCBB* загружается адрес *PCB* для нового процесса и с помощью инструкции *LDPCTX* обеспечивается загрузка аппаратных регистров из *PCB* нового процесса. Так как после выполнения инструкции *SVPCTX* операционная система работает в режиме ядра со стеком прерываний (ввиду отсутствия в этот момент времени выполняемых процессов), то запуск на выполнение нового процесса после загрузки его аппаратного контекста по инструкции *LDPCTX* осуществляется по инструкции *REI*. Поэтому инструкция *LDPCTX*, кроме того должна занести в стек прерывания значения регистров *PC* и *PSL*.

Так как каждый процесс в системе применяет свои стеки и указатели стеков для всех возможных режимов доступа к процессору, то переключение контекста процессов может выполняться даже в случае, когда в контексте процесса выполняется некоторая системная программа в режиме ядра. Это является особенностью архитектуры СМ 1700 в отличие от большинства других ЭВМ, где операционная система всегда выполняется в особом режиме, в котором она не может быть прервана.

Адресное пространство является частью контекста каждого процесса. Управление доступом к памяти процесса осуществляется системными программами и поддерживающей эти программы аппаратурой (контроллером памяти). Управление памятью в операционной системе должно обеспечить для каждого пользователя представление, что он работает в непрерывном пространстве адресов памяти, начиная с нулевого адреса, а также распределение физической памяти между процессами. Кроме того, управление памятью должно контролировать возможность требуемого пользователем режима доступа к памяти (чтение или запись). Для эффективного использования памяти операционная система должна обеспечивать одновременное нахождение в памяти нескольких процессов. Далее рассмотрим, каким образом архитектура СМ 1700 обеспечивает функции управления памятью.

Виртуальное адресное пространство в СМ 1700, определяемое 32-разрядным адресом, представляется для пользователя непрерывным пространством объемом в 2^{32} байт. Виртуальное адресное пространство является настолько большим, что естественно не может быть полностью отображено в физической (оперативной) памяти. Поэтому диспетчер памяти отображает только часть виртуального адресного пространства процесса в физической памяти.

Виртуальное адресное пространство разделяется на две функциональные области: размещения процессов и операционной системы.

Область операционной системы является общей для всех процессов, доступ к этой области из программ пользователя или других обслуживающих программ осуществляется через вызовы системных процедур. Каждый процесс имеет свое собственное адресное пространство, однако несколько процессов имеют возможность обращения к одной и той же области операционной системы. Таким образом, обеспечивается разделение памяти между процессами.

Виртуальное адресное пространство (ВАП) разделяется на части по 512 байт, называемые страницами. Программа строится в виде последовательного набора страниц, пронумерованных от нуля до некоторого верхнего значения. Так как размер страницы 512 байт, то для адресации байта внутри страницы используется 9 бит 32-разрядного виртуального адреса.

Старшие разряды с 9 по 31 виртуального адреса указывают номер виртуальной страницы. Адресация байтов в странице и нумерация номеров виртуальных страниц начинается с 0.

Виртуальное пространство в СМ 1700 разделяется на две области. Младшая половина виртуальной памяти предназначена для размещения процессов (область процессов), старшая половина виртуальной памяти используется системой (область системы). Область процессов в свою очередь делится на две равные части: программную область (P0) и управляющую область процессов (P1).

Старшая половина области системы не используется. Два старших разряда виртуального адреса определяют область, к которой принадлежит адрес: 00—P0-область; 01—P1-область; 10 — область системы; 11 — не используется.

Отображение виртуального адресного пространства в физической памяти осуществляется с помощью контроллера памяти. Виртуальная страничная организация памяти позволяет хранить в физической памяти только часть страниц для программы, а другую часть страниц, которые в данный момент не используются, хранить на диске. Эти страницы будут загружаться в физическую память при обращении к ним, т. е. по мере необходимости, замещая те страницы, которые уже не используются. Такая процедура загрузки страниц с диска называется обменом страниц. Страницы физической памяти, выделяемые для программы, могут быть несмежными, т. е. находиться в разных местах физической памяти. Однако при выполнении программы контроллер памяти должен каждый виртуальный адрес, генерируемый процессором, преобразовать (транслировать) в соответствующий физический адрес. Для трансляции виртуальных адресов в физические используется структура данных, называемая таблицей страниц, которая содержит информацию, необходимую для трансляции адресов. Таблица страниц представляет набор записей, по одной на каждую страницу виртуальной памяти. Каждая запись

имеет размер двойного слова и указывает: режим доступа процессора (чтение или запись) к соответствующей странице физической памяти; расположение страницы — в физической памяти или на диске; номер страницы в физической памяти, если она находится в памяти.

Если данная страница не находится в памяти, запись в таблице страниц определяет адрес образа этой страницы на диске.

Аппаратура контроллера памяти использует таблицу страниц при каждом обращении к памяти. Номер виртуальной страницы в виртуальном адресе используется для нахождения соответствующей записи в таблице страниц. При единичном значении бита достоверности и соответствии кода защиты адрес страницы в физической памяти определяется конкатенацией младших 21 разрядов записи и младших 9 разрядов виртуального адреса.

Операционные системы

Операционные системы ВК СМ 1700 представлены двумя системами: многофункциональной операционной системой, поддерживающей виртуальную память (МОС ВП); диалоговой единой мобильной операционной системой (ДЕМОС-32).

Операционная система МОС ВП

МОС ВП является дальнейшим развитием одной из наиболее распространенных операционных систем 16-разрядных СМ ЭВМ — ОС РВ.

МОС ВП включает в себя следующие основные компоненты: управляющую программу, которую составляют подсистемы ввода-вывода и управления памятью, программа-планировщик, программы системного обслуживания, а также структуры данных, необходимых для функционирования системы; систему управления данными (СУД-32), реализующую процедуру ввода-вывода для файлов на внешних устройствах; интерпретаторы команд оператора, обеспечивающие пользователей возможностями работы с операционной системой путем ввода команд с терминала. МОС ВП включает в себя два стандартных интерпретатора *DCL* и *MCR*, а также содержит средства разработки собственных интерпретаторов; набор системных обслуживающих программ; системы программирования; средства разработки программ пользователя.

Существенной особенностью МОС ВП является выполнение различных компонентов операционной системы в одном из четырех режимов работы процессора. Так, управляющая программа выполняется в режиме ядра, система управления данными — в режиме управления, интерпретаторы команд оператора — в режиме супервизора, системные обслуживающие программы — в режиме пользователя. Компоненты системы, выполняющиеся в менее привилегированном режиме, могут пользоваться средствами более привилегированного. Такая организация повышает надежность и улучшает возможности управления взаимодействием и синхронизацией работы компонентов операционной системы.

Процесс может управлять состоянием других процессов в системе, а также создавать (порождать) новые процессы. Порожденный процесс, получающий собственные значения квот (пределов) используемых разделяемых ресурсов системы, называется отсоединенным. Если же породивший процесс использует значения квот совместно с порожденным, то последний будет называться подпроцессом. Процесс вместе с совокупностью порожденных им подпроцессов называется заданием.

Пользователь МОС ВП может создавать процессы, выполняющиеся в интерактивном и пакетном режимах. Интерактивный процесс создается при регистрации пользователя в системе. Процесс, выполняющийся в пакетном режиме, можно создать, используя стандартный интерфейс пользователя с операционной системой.

МОС ВП управляет отображением виртуального пространства процесса в физической памяти. При этом система гарантирует выбранным для исполнения процессам лишь определенное количество страниц, называемых рабочим набором. Процессы, которым система выделяет рабочие наборы, образует балансный набор. В системе имеются средства изменения размера рабочего набора процесса. Это изменение может выполняться или пользователем, или системой по специальному алгоритму, который учитывает общую активность процессов.

При попытке процесса использовать страницы, отсутствующие в рабочем наборе, осуществляется процедура замещения страниц. Для указанного замещения на дисковом томе используется системный файл для обмена страниц. При выгрузке всего процесса из памяти, страницы его рабочего набора сохраняются в системном файле выгрузки на дисковом томе.

Таким образом, концепция рабочего набора позволяет гибко использовать доступную процессам физическую память. Вся физическая память системы может быть представлена разделенной на три основные части. Одна часть резервируется для ядра операционной системы, другая используется для обмена страниц, третья часть выделяется для рабочих наборов процессов. Выбор процессов, которым предоставляется физическая память и другие ресурсы системы, выполняет программа-планировщик, входящая в ядро операционной системы.

Планировщик обеспечивает диспетчеризацию процессов в системе, а также управляет продолжительностью их выполнения. Система всегда стремится к обеспечению выполнения как можно большего числа процессов. Для достижения этой цели планировщик разделяет центральный процессор между готовыми к выполнению процессами и обеспечивает параллельное выполнение операций. Например, если процесс ожидает завершения операции ввода-вывода, на выполнение запускается другой процесс.

Конкуренция процессов в системе осуществляется на основе приоритетов. Каждому процессу в системе назначается программный приоритет от 0 до 31. Младшие значения приоритетов (от 0 до 15) назначаются обычным процессам, а старшие (от 26 до 31) — процессам реального времени.

Планирование процессов в системе осуществляется с использованием режима круговой диспет-

черизации: процессы получают возможность выполнения на основе системных событий и соответствующих приоритетов и состояний. Выполняемый процесс получает процессор только на интервал (квант) времени, значение которого определяется системным параметром. Получив свой квант времени, процесс выполняется до тех пор, пока не произойдет одно из следующих событий: другой процесс с более высоким приоритетом становится выполняемым; процесс перешел в состояние ожидания; истек текущий квант времени.

Если по истечении кванта времени нет других готовых к выполнению процессов с тем же или высшим приоритетом, текущий процесс получает следующий квант времени.

Планировщик определяет следующие состояния процессов: текущий процесс (*CUR*) является единственным в системе, который в данный момент времени занимает ЦП; готовые к выполнению процессы, находящиеся в оперативной памяти (COM) или на диске (COMO).

Для каждого из состояний COM и COMO, формируются по 32 приоритетные очереди процессов, ожидающих какого-либо системного ресурса или события (всего 11 состояний).

При необходимости выбора на выполнение следующего процесса планировщик проверяет указанные значения приоритетов процессов, которые находятся в состоянии готовности к выполнению в оперативной памяти, и выбирает первый процесс из самой приоритетной очереди, а бывший текущий перемещает в конец соответствующей очереди.

Основными компонентами системы ввода-вывода являются: система управления данными (СУД-32); модули управляющей программы, реализующие процедуру системного обслуживания запроса ввода-вывода QIO; вспомогательные управляющие файловые процессы и драйверы устройств.

Система управления данными обеспечивает для пользователя возможность независимой работы с файлами на внешних устройствах. Эта система состоит из набора подпрограмм, обращение к которым выполняется либо неявно в программах на языках программирования высокого уровня, либо явно с использованием стандартных макрокоманд в программах на языке Макроассемблер.

СУД-32 может быть использована при работе с любыми устройствами, включая терминалы и устройства печати, однако в первую очередь она предназначена для реализации программного интерфейса с устройствами с файловой структурой, такими, как магнитные диски и ленты. При этом СУД-32 поддерживает последовательную, относительную и индексно-последовательную организацию файлов, различные форматы записей и способы доступа к записям в файлах.

СУД-32 реализует логический уровень ввода-вывода (ориентированный на обработку файлов и записей), на котором выполняются операции, независимые от конкретного внешнего устройства.

Процедура обслуживания *QIO* реализует самый нижний, доступный пользователю, физический уровень ввода-вывода (ориентированный на обработку томов и блоков). Она может быть вызвана из программы пользователя или явно или неявно средствами системы управления данными.

Обработка системой запроса *QIO* заканчивается вызовом драйвера устройства программы, непосредственно выполняющей операцию ввода-вывода на устройстве.

При работе с устройствами с файловой структурой вызов драйвера устройства предваряется дополнительной обработкой запроса *QIO*. Эту обработку выполняют вспомогательные управляющие файловые процессы (АСР). В системе МОС ВП реализованы АСР для работы с файловой структурой МОС ВП и ОС РВ на магнитных дисках и лентах, для обработки файловых запросов на сетевое обслуживание и др.

Особенностью драйверов внешних устройств МОС ВП является наличие в них подпрограмм предварительной и заключительной обработки запроса ввода-вывода, выполняющихся в контексте процесса, запросившего ввод-вывод, тогда как собственно работа драйвера выполняется в рамках специального «форк-процесса».

МОС ВП включает в себя средства, обеспечивающие режим совместимости с операционной системой ОС РВ. Реализация программной совместимости МОС ВП и ОС РВ базируется на аппаратной совместимости СМ 1700 и 16-разрядных СМ ЭВМ.

Процессор СМ 1700 может работать в двух режимах: базовом, когда исполняется набор инструкций собственно СМ 1700; режиме совместимости, когда процессор исполняет набор инструкций СМ 1420.

Таким образом, под управлением МОС ВП может быть выполнен образ задачи, подготовленный в ОС РВ, при этом переключение процессора в режим совместимости будет осуществлено операционной системой в момент активизации образа.

Интерфейс пользователя с операционной системой в режиме совместимости осуществляется посредством программной эмуляции операционной среды ОС РВ. При попытке обращения пользователя из задачи, выполняющейся в режиме совместимости, к операционной системе ОС РВ возникает аппаратное прерывание, которое вызывает автоматический переход в базовый режим. Работая в базовом режиме, МОС ВП выполняет необходимые операции по обслуживанию запроса задачи и переключается в режим совместимости для продолжения работы задачи.

Следует заметить, что эмуляция среды ОС РВ в МОС ВП не является полной. Например, не допускается выполнение привилегированных задач с отображением в память управляющей программы ОС РВ, а также задач, использующих директивы управления памятью или непосредственно обращающихся к внешней странице памяти.

Помимо программных средств эмуляции среды ОС РВ операционная система МОС ВП включает программные компоненты ОС РВ, обеспечивающие разработку пользовательских программ, работающих в режиме совместимости: текстовый редактор; программу работы с файлами; транслятор с языка Макроассемблер; построитель задач; процессор косвенных командных файлов и др.

Процедура получения рабочей версии системы МОС ВП из набора дистрибутивных носителей называется установкой системы. В ходе установки происходит копирование необходимых компонентов операционной системы с дистрибутивных носителей на системный диск (при необходимости и на библиотечный диск), определение системных параметров,

создание и инициализация файлов, необходимых для функционирования системы.

Установка заменяет для МОС ВП очень трудоемкую процедуру генерации операционной системы ОС РВ для 16-разрядных СМ ЭВМ.

При этом в отличие от ОС РВ, ядро которой поставляется в виде исходных текстов, а затем в ходе генерации транслируется и компонуется в образ системы, ядро МОС ВП и драйверы устройств на дистрибутивных носителях хранятся в виде образов, готовых к загрузке в оперативную память. Следует заметить, что в ходе установки не выбирается набор возможностей, предоставляемых операционной системой. Эти возможности определены заранее и не подлежат изменению, а выполняется только настройка системы по определенным параметрам. Кроме того, существенным является тот факт, что определение конфигурации внешних устройств для системы выполняются не на этапе установки, а автоматически в процессе загрузки системы.

Установка в системе дополнительных программных продуктов (систем программирования с языков высокого уровня, систем управления базами данных, пакетов прикладных программ), осуществляется путем выполнения специальных командных процедур под управлением операционной системы.

Запуск МОС ВП начинается с выполнения процедуры загрузки ядра операционной системы с дискового устройства в оперативную память. Реализация процедуры загрузки в МОС ВП отличается от аналогичной процедуры для 16-разрядных СМ ЭВМ.

В процессе загрузки МОС ВП участвуют две программы: первичный загрузчик и вторичный загрузчик. Первичный загрузчик загружается в память с кассетной ленты и получает управление в результате выполнения команд консольной подсистемы. Первичный загрузчик ищет на системном дисковом устройстве файл, содержащий образ программы вторичного загрузчика, загружает его в память и передает ему управление. Вторичный загрузчик, который связан непосредственно с файлом образа системы, выполняет операции по непосредственной загрузке и инициализации системы.

Указанная реализация процедуры загрузки преследует несколько целей. Во-первых, программа первичной загрузки превращается в гибкое средство, которое может использоваться не только для загрузки операционной системы, но и, например, для загрузки диагностического супервизора, во-вторых, существенным является факт непосредственной работы первичного и вторичного загрузчика с файловой системой, что позволяет изменить и перемещать файлы, связанные с инициализацией системы (включая файл образа самой системы), выполнять операции переноса операционной системы на диски различных типов.

Инициализация системы после загрузки ядра выполняется модулем ядра операционной системы *INIT* и специальным процессом *SYSINIT*. *SYSINIT* активизирует аппаратуру контроллера памяти и устанавливает содержимое системных структур данных. *INIT* открывает системные файлы: файл выгрузки и файл страничного обмена, порождает системные процессы, включая процесс выполнения начальной командной процедуры.

В ходе работы начальной командной процедуры выполняются следующие основные операции: создаются системные логические имена; устанавливаются привилегированные и разделяемые образы, необходимые для функционирования системы; создаются дополнительные системные процессы (управления очередями, обслуживания системной консолью, сбора статистики, регистрации ошибок оборудования); автоматически определяется конфигурация внешних устройств и выполняется загрузка в память соответствующих драйверов; выполняется пользовательская, начальная командная процедура.

Возможности использования ВК СМ 1700 в различных областях применения обеспечиваются широким набором систем программирования и развитым базовым ПО, выполняющимся в среде МОС ВП.

Под управлением МОС ВП функционируют следующие системы программирования: ФОРТРАН, КОБОЛ, ПАСКАЛЬ, БЕЙСИК, СИ, БЛИСС, ПЛ/1, Модула-2, КОРАЛ, ЛИСП, ПРОЛОГ, АДА.

Операционная система ДЕМОС-32

ДЕМОС-32 разработана в рамках проекта по созданию единой операционной среды (ЕОС) ДЕМОС. Основной задачей проекта ЕОС является обеспечение единого интерфейса пользователя и программиста для ЭВМ различных архитектур. При этом ставится условие обеспечения совместимости указанного интерфейса с интерфейсом широко распространенной операционной системы *UNIX*.

По своему назначению операционная система ДЕМОС является системой разделения времени с возможностью выполнения пакетных заданий. Средства обеспечения режима реального времени в основной состав системы не входят. Система ДЕМОС имеет следующие основные особенности: иерархическая древовидная схема управления процессами, в которой каждый процесс имеет свое адресное пространство, которое может быть использовано любым доступным образом (отсутствуют зарезервированные области для системных данных в виде таблицы, буферов); отсутствие встроенного интерпретатора команд пользователя (практически любая программа может быть интерпретатором команд, причем для каждого пользователя своя); иерархическая древовидная файловая система неограниченной глубины; унифицированное представление файлов и периферийных устройств, что позволяет использовать одни и те же программы, как для обычных файлов в файловой системе, так и для периферийных файлов (устройств); отсутствие ограничений на представление данных в файлах (отсутствуют блоки записей, при этом любой файл рассматривается как пронумерованная последовательность байтов, что обеспечивает эффективную работу с файлами); согласованное представление входных и выходных данных для программ позволяет создавать «поточные линии» по обработке данных без использования промежуточных временных файлов; полный комплект программных средств для создания и обработки текстов, таблиц с большими и маленькими русскими и латинскими буквами; наличие специальных средств описания терминалов позволяет ликвидировать зависимость

программных средств (экранных редакторов и т. д.) от конкретных возможностей оборудования.

По своим функциональным характеристикам ДЕМОС-32 является расширением системы ДЕМОС для 16-разрядных СМ ЭВМ, что обусловлено в первую очередь архитектурой СМ 1700 и набором ее внешних устройств. К отличительным особенностям системы ДЕМОС-32 следует отнести: механизм страничного управления виртуальной памятью, что позволяет увеличить количество активных процессов в системе и уменьшить объем подкачки процессов; 32-разрядная архитектура позволила снять ряд ограничений на размер ядра системы, что в свою очередь улучшило функциональные характеристики, снизило затраты на работу самой системы и расширило функции ядра; расширение адресного пространства позволило улучшить качественные и количественные характеристики программных компонентов системы (например, максимальный размер обрабатываемых файлов, программ и т. д.); новая организация файловой системы (при сохранении ее логической структуры) на внешних носителях позволила на порядок поднять пропускную способность файловой системы, что является немаловажным фактором при обработке больших массивов данных, повысить надежность хранения информации на внешних носителях.

Текущая версия системы ДЕМОС-32 предназначена в основном для создания инструментальных систем по разработке больших программных проектов, на ее основе могут быть созданы информационные, обучающие системы, системы подготовки документации, системы распределенной обработки информации.

В последующих версиях системы ДЕМОС-32 планируется расширить поддержку внешних устройств, новых моделей 32-разрядных СМ ЭВМ. Будет достигнута большая совместимость с фактическим мировым стандартом *UNIX System V*. В состав системы будут включены реляционная СУБД, графические средства на базе ГКС, поддержка сетей, новые системы программирования.

Системы управления базами данных

ПО для организации и управления базами данных включает многофункциональную информационную систему (МИС СМ) и комплексную автоматизированную реляционную систему (КАРС).

Многофункциональная информационная система (МИС СМ)

МИС СМ — ПО для централизованного управления базами данных, интерактивной и пакетной обработки данных на СМ 1700 под управлением операционной системы МОС ВП. Пользовательские данные хранятся либо в файлах операционной системы МОС ВП, либо в виде базы данных сетевой структуры, управляемой системой «Сеть-32». Доступ к данным возможен либо из программ на языках программирования (Макро, КОБОЛ, БЕЙСИК, ФОРТРАН, ПАСКАЛЬ, ПЛ/1, СИ и др.), либо через интерактивную систему запросов.

МИС СМ состоит из четырех программных компонентов, которые могут использоваться как отдельно, так и совместно: система управления словарями («Словарь-32»); система управления базами данных («Сеть-32»); система управления формулярами (СУФ-32); интерактивная система запросов (ФОБРИН-32).

СИСТЕМА УПРАВЛЕНИЯ СЛОВАРЯМИ «Словарь-32» предназначена для организации и ведения централизованного словаря данных. Все описания данных (метаданные) компонентов МИС СМ, а также описания файлов на языке *SLVL* хранятся в словаре данных. В словаре хранятся следующие типы метаданных: описания структуры файлов на языке *SLVL*. Описание записи на *SLVL* может быть скопировано в программу на любом из языков программирования (КОБОЛ, БЕЙСИК, ФОРТРАН, ПАСКАЛЬ, ПЛ/1, СИ) для обработки одних и тех же файлов программами на нескольких языках; описания записей, доменов, таблиц и процедур системы ФОБРИН-32; описания схем, подсхем, схем управления доступом и схем хранения СУБД «Сеть-32».

Словарь организован в виде иерархической структуры каталогов и объектов. Объекты находятся на самом нижнем уровне иерархии и содержат метаданные.

Авторизация метаданных осуществляется с помощью таблиц управления доступом. Обслуживающие программы «Словарь-32» выполняют необходимые функции по поддержке целостности метаданных (защита и восстановление, проверка структуры, сбор статистической информации об изменении объектов).

СИСТЕМА УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ «Сеть-32» реализует предложения комитета КОДАСИЛ по базам данных. Языки описания данных (ЯОД) «Сеть-32» позволяют эффективно реализовать сложные логические взаимосвязи данных пользователей, обеспечивают высокую реактивность прикладных программ, совместный доступ многих пользователей к базам данных, высокую степень защиты данных от разрушения и неавторизованного доступа.

«Сеть-32» поддерживает одновременную работу 60 пользователей на одной ЭВМ с различными базами данных. Целостность данных обеспечивается средствами полного и частичного копирования баз данных, ведения журналов изменений и механизма захватов.

Прикладные программы могут обращаться к базам данных при программировании на любом включающем языке, причем язык Кобол включает операторы языка манипулирования данными (ЯМД) «Сеть-32», а для Фортрана имеется прекомпилятор ЯМД. Для остальных языков (Макро, ПАСКАЛЬ, ПЛ/1, СИ, БЕЙСИК) доступ к базам данных осуществляется через *CALL*-интерфейс. Для отладки прикладных программ, а также интерактивного взаимодействия с базами данных в состав «Сеть-32» входит интерактивная подсистема запросов *DBQ*. Особенность *DBQ* является возможность автоматического отображения структуры ис-

пользуемой подсхемы на видеотерминале в виде диаграммы Бахмана. Доступ к базам данных «Сеть-32» возможен и через интерактивную систему запросов ФОБРИН-32.

СИСТЕМА УПРАВЛЕНИЯ ФОРМУЛЯРАМИ СУФ-32 представляет собой набор программных средств для создания, хранения и использования в прикладных программах экранных формуляров (видеоформ), имеющих вид, максимально приближенный к виду документов (анкеты, бланки и т. п.). Видеоформа состоит из постоянной информации (заголовки, рамки и т. д.) и полей для ввода и отображения данных.

СУФ-32 позволяет задавать контроль данных, вводимых пользователем с терминала в поле, видеоатрибуты полей (реверс, яркость, мерцание), использовать символы двойной высоты и ширины, наложение видеоформ друг на друга.

Видеоформы СУФ-32 могут быть использованы в программах на всех языках программирования, а также при работе с данными через интерактивную систему запросов ФОБРИН-32.

ИНТЕРАКТИВНАЯ СИСТЕМА ЗАПРОСОВ ФОБРИН-32 — интерактивный процесс запросов, позволяющий формировать запросы к данным на специальном языке высокого уровня, ориентированном как на программистов, так и непрограммистов.

Универсальность ФОБРИН-32 заключается прежде всего в том, что он позволяет обрабатывать данные, хранящиеся как в базе данных СУБД «Сеть-32», так и в СУД-файлах последовательной, относительной и индексной организации, используя при этом унифицированный язык запросов.

Описания данных для ФОБРИН-32 могут быть написаны либо на собственном ЯОД системы, либо на ЯОД системы «Словарь-32» (*SLVL*). Эти описания, а также процедуры для обработки данных хранятся в централизованном словаре, обслуживаемом системой «Словарь-32».

ФОБРИН-32 имеет в своем составе генератор отчетов, позволяющий формировать сложные форматированные отчеты. Для начинающих пользователей предусмотрены специальный режим сопровождения и средство генерации описаний, которые с помощью подсказок и «меню» позволяют быстро освоить ЯОД и ЯМД ФОБРИН-32.

При обработке данных (модификация, поиск) могут использоваться экранные формы, подготовленные с помощью системы СУФ-32. *CALL*-интерфейс ФОБРИН-32 позволяет разрабатывать прикладные программы на языках (ФОРТРАН, КОБОЛ, БЕЙСИК, ПАСКАЛЬ и др.), расширяющие возможность ФОБРИН-32 по обработке данных.

Комплексная автоматизированная реляционная система (КАРС)

КАРС обеспечивает хранение и выборку алфавитно-цифровой информации для обработки в различных областях применения.

КАРС поддерживает реляционную структуру данных: представление в виде отношений (таблиц), состоящих из строк и столбцов. В системе реализован язык *SQL*, который является наиболее мощным из языков, используемых в реляционных СУБД. Помимо ядра системы, интерпретирующего команды *SQL*, КАРС включает в себя: системный интерактивный интерфейс (СИН), позволяющий выполнять команды *SQL* в интерактивном режиме и управлять форматированием выходной информации; систему интерактивных приложений (СИП), обеспечивающую обработку данных через экранные формы, ориентированную на конечного пользователя; генератор отчетов, выполняющий форматирование и вывод информации из базы данных с включением вспомогательного текста; обслуживающие программы, обеспечивающие загрузку, реорганизацию и восстановление базы данных; интерфейсы с языками программирования высокого уровня.

КАРС обеспечивает мультидоступ к данным с защитой данных от одновременного обновления, от неавторизованного доступа. Все изменения данных заносятся в файл изменений, что обеспечивает возможность восстановления базы данных после аппаратных сбоев.

Физическая структура базы данных обеспечивает эффективное хранение данных и вместе с тем возможность быстрого поиска. Для этого применяются индексирование и кластеризация таблиц.

База данных КАРС может размещаться в нескольких файлах, что позволяет более гибко использовать физическое пространство внешних запоминающих устройств.

Простота взаимодействия с системой КАРС, наличие компонентов, ориентированных на конечного пользователя, позволяют существенно сократить сроки освоения системы и ускорить создание на ее основе информационных систем.

Программные средства распределенной обработки данных

В состав программных средств распределенной обработки данных входят: система ПО распределенных сетей (СПО ТРАЛ); система ПО локальных сетей (СПО МАГИСТР); система ПО для организации распределенных многомашинных компексов на базе СМ 1700 и ЕС ЭВМ (СПО ЭМУЛЯТОР); система ПО для сетей СМ ЭВМ с малыми ресурсами (СПО СЕТЬ МИНИ); однородная операционная среда ОС ДЕМОС для сети ЭВМ различных типов (СПО ДЕМОН); система ПО локальных сетей кольцевого типа (СПО КОЛОС).

СПО ТРАЛ является базовым сетевым ПО для создания сетей ЭВМ на базе СМ 1700. СПО ТРАЛ позволяет создавать однородные распределенные сети ЭВМ СМ 1700 любой топологии и размерностью до 1024 узлов. Использование пакета программ для создания однородных сетей СМ ЭВМ (ПП СЕТЬ СМ) обеспечивает включение ЭВМ типа СМ 1420 в качестве узлов в создаваемую распределенную сеть. СПО ТРАЛ работает под управлением операционной системы МОС ВП и обеспечивает следующие основные функции: взаимодействие пользовательских программ, выполняемых в различных узлах сети; доступ из программ и с тер-

миналов к файлам в удаленных узлах; обмен файлами между различными узлами сети; управление удаленными программами; терминальные взаимодействия; адаптивное управление потоками данных и функционированием сети в целом; использование терминалов, подсоединенных к различным узлам, как сетевые; тестирование телекоммуникационного оборудования и программного обеспечения.

Связь между узлами обеспечивается асинхронным телекоммуникационным оборудованием из номенклатуры СМ ЭВМ.

СПО МАГИСТР является базовым сетевым ПО для создания на базе ВК СМ 1700 локальных сетей магистрального типа с множественным доступом, с обнаружением несущей и наложений со скоростью передачи данных до 10 Мбит/с.

Информационно и локально СПО МАГИСТР совместима с СПО ТРАЛ и допускается их объединение. Существенным новым фактором, расширяющим возможности системы, является использование быстродействующей магистрали для организации информационной связи и создания локальной сети. Высокая скорость передачи обеспечивает качественно новый уровень взаимодействия ЭВМ.

СПО ЭМУЛЯТОР предназначена для организации распределенных многомашинных комплексов на базе ЕС ЭВМ и СМ 1700. Путем эмуляции терминальной станции ЕС 7920 в СПО ЭМУЛЯТОР обеспечивается функционирование СМ 1700 в качестве удаленных абонентских пунктов ЕС ЭВМ и обмен данными между программами в СМ и ЕС ЭВМ.

СПО СЕТЬ МИНИ предназначена для организации сетевого взаимодействия разнотипных СМ ЭВМ, функционирующих под управлением различных ОС. Комплекс программ, составляющих СПО СЕТЬ МИНИ, функционирует на СМ 1700 под управлением МОС ВП, на СМ 1420 под управлением ОС РВ и РАФОС, на СМ 1800 — ОС 1800, на СМ 1810 и ЕС 1840 — МЛОС.

Система ПО СЕТЬ МИНИ для организации взаимодействия ЭВМ использует терминальные линии связи и обеспечивает: терминальный доступ к удаленным узлам, т. е. установление логической связи терминалов одной ЭВМ с другими узлами сети; обмен файлами между узлами сети по командам оператора.

СПО ДЕМОН предназначена для использования в качестве базовой сетевой среды при создании сетей на базе СМ 1700 и СМ 1420, работающих под управлением ОС ДЕМОС. В этой операционной среде обеспечивается обмен файлами между узлами сети и терминальный доступ к удаленным узлам.

СПО КОЛОС предназначена для использования в качестве базового сетевого ПО при создании локальных сетей кольцевого типа из разнотипных СМ ЭВМ и оконечного оборудования (терминалы, печатающие устройства). ЭВМ и оконечное оборудование включаются через станции локальной сети кольцевого типа (СЛК-СМ) в кольцевую локальную сеть. Объединяемые в сеть СМ ЭВМ могут работать под управлением различных ОС. При этом обеспечивается реализация следующих основных функций: межмашинное взаимодействие до 125 ЭВМ со скоростью передачи по кольцу 500 000 бит/с, а между СЛК-СМ и присоединенным оборудованием до 19 200 бит/с; обмен файлами между узлами локальной сети по командам оператора; терминальный доступ к удаленным узлам; сетевой терминальный доступ, т. е. организация логической связи сетевой СМ ЭВМ с терминалом, подключенным непосредственно к СЛК-СМ.

Программные средства машинной графики

К программным средствам машинной графики и САПР относится прежде всего базовое ПО автоматизированных рабочих мест на базе СМ 1700 (БПО АРМ СМ 1700).

БПО АРМ СМ 1700 предназначено для широкого применения в САПР и обеспечивает работу пользователей в режиме реального времени и пакетном. БПО АРМ СМ 1700 включает в себя базовые средства машинной графики, средства поддержки проблемно ориентированных графических пакетов, один из графических проблемно ориентированных пакетов и драйверы графических устройств, входящих в состав комплекса АРМ СМ 1700.

БПО АРМ СМ 1700 ориентировано на работу в операционной системе МОС ВП.

Базовые программные средства машинной графики (БПС МГ) на основе графической корневой системы (ГКС), предназначены для поддержки графических устройств, функционирующих в составе комплекса СМ 1700.

Отличительной особенностью БПО АРМ СМ 1700 является его проблемная ориентация путем включения в состав БПО одного из проблемно ориентированных пакетов программ: автоматизированного проектирования БИС; автоматизированного проектирования изделий машиностроения; автоматизированного проектирования сложных поверхностей, включающих программы проектирования изделий для авиа-, авто- и судостроения с выдачей результатов проектирования на станки с ЧПУ; автоматизированного архитектурно-строительного проектирования (в двух- и трехмерном изображении) архитектурных и строительных конструкций, зданий, сооружений и др.; прочностных расчетов методами конечных элементов, включающих программы прочностных расчетов в интерактивном режиме с графическим отображением конструкций; автоматизированного проектирования печатных плат.

Система программного диагностирования ВК СМ 1700

Система программного диагностирования ВК СМ 1700 с учетом структуры центральной части ВК СМ 1700 реализована по принципу расширяющихся областей (сред) диагностирования. Среда диагно-

стирования характеризуется совокупностью аппаратных и программных средств, предоставляющих диагностическим программам определенные функциональные возможности по организации процесса диагностирования и выбору тестовых воздействий. Принцип расширяющихся областей в настоящее время является наиболее эффективным принципом реализации процедуры самодиагностирования для ВК. Согласно этому принципу в рамках каждой среды диагностирования ВК СМ 1700 более низкого уровня проверяется аппаратное ядро, т. е. необходимый минимум работоспособной аппаратуры среды более высокого уровня. Это позволяет не только упростить процесс поиска неисправностей, но и в значительной степени избежать влияния некоррективных неисправностей на этот процесс.

Программные средства диагностирования имеют иерархическую структуру, в которой можно выделить шесть уровней (1, 2, 3, 4, 5 и 6). Диагностические программы каждого более высокого уровня требуют для своего запуска и выполнения большего аппаратного ядра и расширяют область аппаратуры, используемой для организации процесса диагностирования.

Первый уровень программной диагностики представлен пакетом программ, выполняемых под управлением операционной системы, который обеспечивает комплексную проверку работоспособности внешних устройств и основных программных компонентов системы.

Следующие пять уровней образуют систему диагностических программ для проверки работоспособности и поиска неисправностей технических средств. На этих уровнях решается задача обеспечения максимальной полноты проверки и детализации задачи поиска неисправностей. При этом детализация достигается за счет взаимной увязки различных диагностических программ по уровням и внутри одного уровня, а также за счет разбиения каждой диагностической программы на тесты, подтесты и выделения в них элементарных проверок.

Диагностические программы уровней 2, 3 и 4 выполняются под управлением диагностического супервизора (ДС). Различие диагностических программ этих уровней связано с использованием ими ресурсов операционной системы МОС ВП.

ДС реализует командный язык, который позволяет выполнить широкий набор функций: настройку системы на заданное соединение проверяемых устройств; загрузку диагностических программ с различных носителей; запуск и управление выполнением диагностических программ; отладку и модификацию загруженных в память программ; выдачу справочной информации о системе программного диагностирования и о составе поддерживаемых ее технических средств.

Создание ДС с возможностями, близкими к возможностям операционных систем, имеет важное значение для повышения эффективности системы программного диагностирования в целом. За счет представления широкого набора сервисных функций со стороны ДС уменьшается среднее время поиска неисправности и время профилактических осмотров. За счет вынесения общих для всех диагностических программ процедур в ДС значительно экономятся затраты на их разработку и унифицируется управление отдельными тестовыми проверками.

Унификация управления наряду с гибкостью позволяет организовать различные специальные режимы проведения диагностирования, например, посредством многократных зацикливаний отдельных групп элементарных проверок (тестов, подтестов) для повышения полноты диагностирования в классе перемежающихся неисправностей, а также для применения диагностических программ на заключительных этапах наладки ВК, а разрешенный в ДС режим командного выполнения диагностических программ — детализировать момент возникновения ошибки.

Диагностические программы уровня 2 выполняются только с использованием операционной системы (прежде всего системы драйверов) и предназначены для проверки работы внешних устройств в мультипрограммном режиме, а также правильности передачи данных с учетом различных протоколов, которые поддерживаются в системе. Важной характеристикой уровня 2 является возможность выполнения процедуры диагностирования с помощью программ данного уровня параллельно с другими процессами в системе. Это позволяет не останавливать работу ВК на время профилактических осмотров и поиска отдельных неисправностей, не нарушающих работоспособности вычислительной системы в целом. Выполнение проверки уровня 2 на завершающих этапах диагностирования СМ 1700 обеспечивает работоспособность технических средств пользователя.

Диагностические программы уровня 3 выполняются как с использованием ресурсов операционной системы, так и без них, под управлением автономно работающего диагностического супервизора. Первый режим работы предоставляет возможность так же, как и на уровне 2, выполнять процедуру диагностирования как пользовательский процесс, не останавливая работу машины. Второй режим предусмотрен на случай, если возникшие неисправности не позволяют запустить операционную систему. В первом случае при выполнении диагностических программ используются системные драйверы, а во втором — драйверы ДС (диагностические драйверы).

Диагностические программы уровня 3 используются для тренировки ЦП и функциональной проверки внешних устройств. Они завершают диагностирование среды системы и совместно с программами уровня 2 подготавливают работу среды пользователя.

Диагностические программы уровня 4 выполняются под управлением автономного ДС. Эти программы осуществляют комплексные проверки: ЦП, привилегированных инструкций, различных режимов ЦП, а также основных функций внешних устройств. На уровне 4 возможен доступ к периферийным устройствам на уровне регистров, что обеспечивает максимальную разрешающую способность при диагностировании, необходимую для ремонта периферии.

Диагностические программы уровня 5 выполняются автономно без ДС. На этом уровне обеспечиваются только самые простые средства загрузки, управления и формирования сообщений об ошибках. Поэтому, несмотря на то, что режим работы уровня 5 представляет широкие возможности для обнаружения неисправностей, диагностические программы этого уровня используются ограниченно,

только для получения такой информации, которую нельзя получить при диагностировании на других уровнях. В настоящее время программы уровня 5 ограничены проверкой набора инструкций, необходимого для работы диагностического супервизора.

Диагностические средства уровня 6 представлены системой диагностических микропрограмм, выполняемых под управлением диагностического микромонитора, и ПЗУ — резидентным микротестом, запускаемым при включении питания.

На этом уровне решаются следующие две важнейшие задачи диагностирования СМ 1700: минимизация аппаратного ядра, необходимого для организации процесса самодиагностирования, и достижение максимальной разрешающей способности.

Диагностические микропрограммы, управляемые микромонитором, делятся на две группы. Программы первой группы запускаются из ОЗУ консольно-диагностического процессора (КДП) и выполняют проверку аппаратуры ЦП, необходимой для работы второй группы программ. Диагностические микропрограммы второй группы запускаются из управляющей памяти АЛП и проверяют другие составляющие ЦП. Такое разбиение позволяет дополнительно уменьшить минимально необходимое аппаратное ядро для начала процесса самодиагностирования.

Значение микромонитора для диагностических микропрограмм аналогично значению ДС для диагностических программ уровней 2, 3 и 4. Микромонитор предоставляет широкий набор сервисных функций по загрузке, запуску и управлению выполнением микропрограмм, по модификации содержимого микропрограммно доступных схем ЦП, позволяет организовать выполнение группы тестов, с возможностью зацикливания, обеспечивает выполнение микропрограмм по микрокомандам.

ПЗУ — резидентный микротест, который выполняется КДП, определяет самый начальный этап самодиагностирования вычислительного комплекса СМ 1700, поскольку не требует средств загрузки микропрограмм. В блоке ЦП имеется переключатель разрешающий отключение микротеста.

Реализация резидентных тестов самодиагностирования представляет собой эффективное средство проверки внутренних компонент комплекса, при минимальном влиянии со стороны других составляющих комплекса. Организованный таким образом доступ к внутренним точкам объекта диагностирования позволяет увеличить полноту и разрешающую способность средств программного диагностирования.

Для удобства выполнения процедуры диагностирования ВК СМ 1700 может быть использована программа «Экспресс-проверки». Эта программа является специальной управляющей программой, позволяющей пользователям выполнять диагностирование комплекса с использованием диагностических и микродиагностических программ. Программа «Экспресс-проверки» может выполняться в автоматическом режиме без вмешательства пользователей или с использованием упрощенного диалога, где можно в режиме «меню» выбрать один из возможных способов диагностирования комплекса.

Кроме рассмотренных средств программной диагностики, определенное место в процедуре диагностирования ВК СМ 1700 занимают программные средства уровня 1 для проверки функционирования комплекса под управлением операционной системы МОС ВП. Эти программные средства предоставляют возможность регистрации аппаратных и программных сбоев в системе, обеспечивают сбор статистики о функционировании системы, позволяют проводить анализ аварийного и текущего состояния системы.

Особое место среди программных средств занимает пакет программ комплексной проверки функционирования вычислительной системы СМ 1700 «Эксперт», который используется как первый уровень системы программного диагностирования. Проверка системы с помощью пакета «Эксперт» не дает исчерпывающей информации об исправности системы в целом, однако успешное выполнение проверок гарантирует готовность системы к эксплуатации.

Программы, входящие в пакет, создают имитацию нагрузки в системе, обеспечивая комплексную проверку работоспособности внешних устройств и основных компонентов системы. Пакет можно применять для предварительного диагностирования комплекса, а также для демонстрации системных возможностей.

Развитие системы программного диагностирования предусматривает построение специальных центров обслуживания, оснащенных экспертными системами для автоматизации поиска неисправностей и связанных линиями связи с удаленными ВК СМ 1700.

СОДЕРЖАНИЕ

Редактор *Т. И. Петрова*

Техн. редактор *М. Я. Орехов*

Корректоры *Л. В. Селиверстова, В. А. Агеева*

Сдано в набор 09.08.88. Подп. в печать 05.12.88. Т—22614 Формат 60×90¹/₈
Бумага типографская № 2. Гарнитура литературная. Печать высокая. Усл. печ. л. 4,0.
Уч.-изд. л. 4,9. Тираж 10083 экз. Заказ № 2094. Изд. № ГСП-265. Цена 75 коп.

Всесоюзный научно-исследовательский институт информации
и экономики (ИНФОРМПРИБОР)
125877, ГСП, Москва, А-252, Чапаевский пер., 14

Типография ВНИИТЭМР, 142002, г. Щербинка Московской обл., ул. Типографская д. 10.

75 к.