КОНСТРУКЦИЯ И СИМУЛЯЦИЯ БЛОКОВ МИКРОПРОГРАММНЫХ УПРАВЛЕНИЙ

П. ХАНТОШ

Кафедра Автоматизации Будапештского Технического Университета.

Представлено проф. д-р. Ф. Чаки

Поступило 1 февраля 1974 г.

Введение

Устройства микропрограммного управления имеют большое преимущество с точки зрения конструкции вычислительных машин. Первый раз слово «микропрограммирование» появилось в 1950-ом году в одной из публикаций Лаборатории им. Линколна. Принцип, изложенный в 1951-ом году М. В. Вильксом, словно революционизировал вычислительную технику.

Современный принцип требует современных методов конструкции. Симуляция на ЭВМ является важным средством автоматизированного проектирования. Суть симуляции заключается в построении имитационных моделей. Теория и метод моделирования распространяется в областях науки, а также в решении конкретных проблем отдельных специальных наук.

В связи с оригинальной моделью Вилькса появилось очень много публикаций, и многие занимались усовершенствованием решения, предложенного им. Несмотря на это, по сути дела модель действительна и сегодня, и в основных принципах её построения существенных изменений нет.

В нашей статье мы занимаемся вопросами проектирования устройств микропрограммного управления. Помимо традиционных средств (схем, кривых времени, и т. п. . . .) большое внимание уделяется описанию излагаемых систем на символических языках. Символические языки имеют значение не только с точки зрения дидактики, но с помощью различных языков, применяемых для симуляции, можно производить анализы с целью синтеза. Некоторые особенно развитые языки могут быть применены и при прямом проектировании электронных систем. Различные микропрограммные структуры и составление микропрограмм — хотя это важные вопросы, — здесь не рассматриваются. В этих областях также распространяются описания на различных специальных языках.

Микропрограммное управление

В настоящее время при исследовании структуры электронных устройств массового производства можно установить следующие: существует несколько функциональных подсистем, выполняющих различные специальные задачи

(например, хранение информации в вычислительной машине, арифметические и логические действия, внутреннюю передачу данных и т. п. . . .).

Устройство выполняет свою задачу в результате совместной работы подсистем. В сжатом виде это задаётся алгоритмом работы устройства. В алгоритме работы описывается, какие операции должны быть выполнены отдельными подсистемами в данные моменты времени. Подсистемы сами по себе не годятся для выполнения функции всего устройства.

Обеспечение выполнения алгоритма работы — это и есть задача устройства управления. Упомянутый ранее алгоритм работы является слишком общим для того, чтобы на его основе устройство управления выполняло регулирование работы подсистем. При проектировании отдельных подсистем для каждой из них можно задать несколько управляющих сигналов четко определенной функции, с помощью которых можно осуществлять управление подсистемой.

Точно так же для отдельных подсистем можно определить сигналы, которые могут влиять на ход выполнения алгоритма работы. Это — переменные, сигнализирующие состояние системы, которые в ходе выполнения алгоритма образуют условия перехода. Определив для каждой подсистемы управляющие сигналы и сигналы состояния, на основе алгоритма работы можно написать так называемый алгоритм управления, что означает запись алгоритма работы в виде последовательности сигналов. Этот алгоритм работы уже может действовать, его можно построить как отдельный блок, поскольку необходимо обеспечить только некоторые предписанные последовательности сигналов.

Вот несколько обычных способов физической реализации:

а) синхронно- или асинхронно-последовательная сеть;
б) устройство управления на фазовых регистрах;
в) применение записывающего счетчика.

Однако, при такой реализации можно спроектировать устройство управления лишь очень ограниченных габаритов. Обеспечение нескольких тысяч операционных шагов и более 100 управляющих сигналов требует сетей огромных габаритов, которые со всех точек зрения являются трудно управляемыми. Их проектирование, построение, засечка и эксплуатационный контроль производства довольно тяжело.

Последующие изменения в алгоритме работы представляют собой отдельную проблему. Конструкторы занимались этими вопросами уже с момента спроектирования первых вычислительных машин.

Одно из решений проблемы, которое дейтсвительно и сегодня, было предложено М. В. Вильксом в 1951-ом году на конференции по вычислительной технике, проведенной в Менчестерском Университете [35]. Придерживаясь к предложенному принципу — с некоторыми изменениями, — в статье,

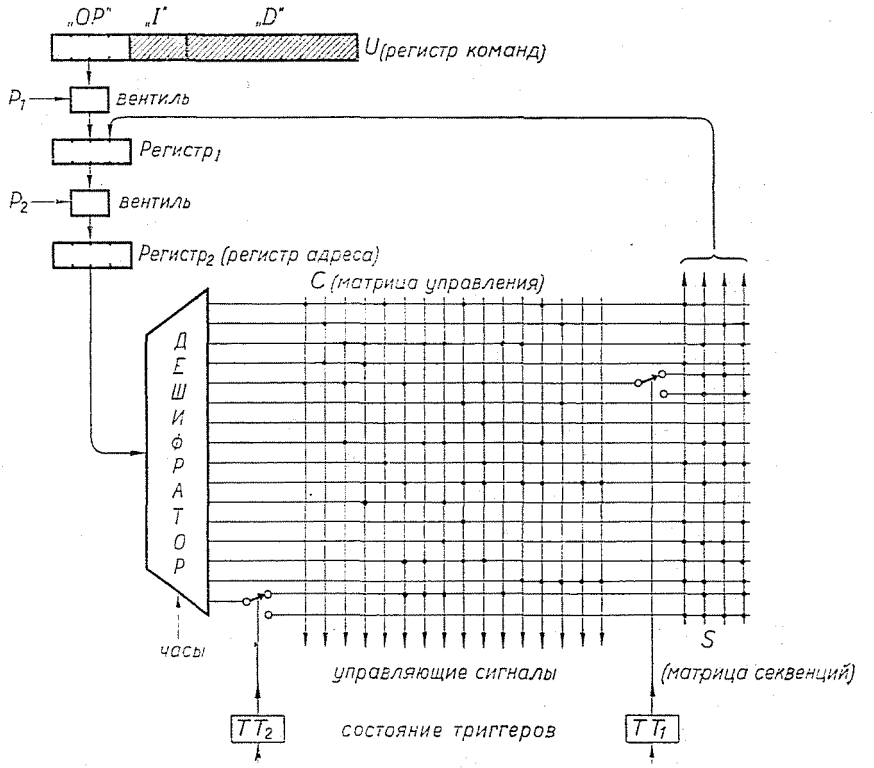


Рис. 1. Гипотетический блок управления

написанной в 1958-ом году [36], он задал схему управления, обеспечивающую возможность систематической перестройки составных частей ЭВМ.

Суть решения состоит в том, что для хранения последовательности управляющих сигналов применяется нондеструктивное, т. е. только читаемое устройство хранения, которое обеспечивает и правильный порядок последовательности сигналов (Рис. 1.). Значит, упрощение конструкции было достигнуто только за счёт увеличения затрат на техническое обеспечение ЭВМ.

Любая машинная команда разбивается на ряд элементарных операций. Например, суммирование двух чисел:

— перенос чисел из регистра в сумматор;
— суммирование;
— перенос суммы из сумматора в регистр.

Этими переносами можно управлять только с помощью активизации вентилей, регулирующих поток информации. Вентили возбуждаются под влиянием управляющих сигналов (сигнальными проводами).

Состояние сигнального провода определяется одним из столбцов матрицы запоминающего устройства.

Матрица представляет собой нондеструктивное, «только читаемое» устройство хранения. Если горизонтальные проводы в матрице возьмём выбирающими и задаём сигнал одновременно только на один провод, тогда по вертикальным проводам появляется информация, записанная в указанной строке матрицы. Значит, состояние управляющих сигналов определяется указанной строкой матрицы.

Дешифратором осуществляется дешифровка n-разрядного выходного сигнала регистра R_2. С помощью n разрядов можно выразить 2^n различных выходов и, соответственно n-разрядному коду можно подавать сигнал на какой-то определенный выход.

Контрольная матрица «С», в которой записаны управляющие сигналы, обладает 2^n горизонтальными и m вертикальными проводами. На рисунке чёрная точка при пересечении проводов символизирует определенное импедансное включение. Сигнал, поданный на горизонтальный провод, появляется и в вертикальном проводе там, где импедансное включение пропускает его.

Выходной сигнал дешифратора активизирует кроме матрицы «С» и соответствующий горизонтальный провод в матрице секвенций «S». Матрица «S» обладает 2^n горизонтальными и n вертикальными проводами, с её выхода получается микрокоманда, выполняемая в следующем цикле. Для выполнения различных способов адресации, различных операций, отскоков и тестов необходимо осуществление условного периода. Одно из возможных решений заключается в исследовании двух устойчивых состояний системы. Выход «I»/«O» зависит от результата навперёд определенного осмотра. Горизонтальная линия, при которой нужно провести осмотр условий, разделяется на две линии в матрице «S», а зависимость следующего адреса от состояния триггеров («I»/«O») обеспечивается переключателем.

Важнейшие наименования:

1. Горизонтальные проводы называются микрокомандами.

2. Число горизонтальных проводов, выходящих из дешифратора, равняется числу микрокоманд, эписываемых в устройство хранения.

3. Число вертикальных проводов ($C + S$ выходов) равняется длине слова.

4. Запоминающее устройство является «только читаемым», поэтому оно называется «**ROM**» (**read only memory**) или «**ROS**» (**read only storage**).

Начальный адрес подпрограммы в микрокомандах задаётся извне, устройство управления получает его из кода операции машинной команды или из части кода. Запись и перенос в регистры управляется сигналами R_1 и P_2. В R_1 содержится адрес выполняющейся микрокоманды, а в R_2 — адрес следующей микрокоманды. (Это содержится в выполняющейся микрокоманде в матрице S.) В начале следующего цикла сигнал P_2 перебросит

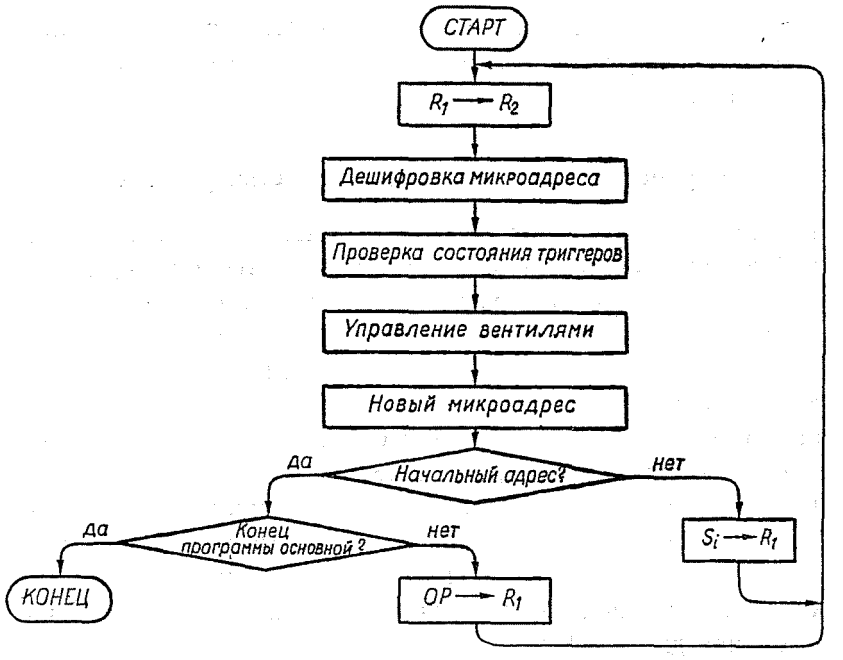


Рис. 2. Функциональная схема последовательности операций

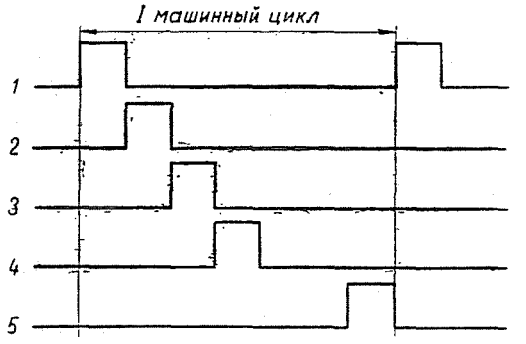


Рис. 3. Временные отношения. 1. Вызов микрокоманды из памяти «РОМ»; 2. Управление занесением в арифметическое-логическое устройство; 3. Арифметическая или логическая операция; 4. Управление выводом из арифметического-логического устройства; 5. Вызов адреса новой микрокоманды

содержимое R_1 в R_2. Под влиянием сигнала P_1 записывается адрес следующей микрокоманды. Схема работы системы представлена на рисунке 2. В более сложных электронно-вычислительных устройствах управляющие сигналы должны обеспечить и одновременное выполнение многих работ. В [17] автор рассматривает временные соотношения устройств микропрограммного управления в общих чертах. В случае простейшего процессора важнейшие фазы

времени показаны на рисунке 3. Описанные фазы должны следовать друг за другой безусловно в предписанном порядке. Их чередование или перекрытие не допускается.

Симуляция гипотетического устройства управления

Использованные в предыдущей части методы описания структуры и работы больших электронно-вычислительных систем (схемы, графики времени, словесное описание) характеризуют систему обычно только с одной стороны. С их помощью построение системы и проверка её работы осуществляются довольно трудно. Однако, для этой цели весьма пригодны высокоразвитые символические языки [7]. Вообще ими можно пользоваться для описания алгоритмов и логических систем. Основы симуляции на символических языках разработаны К. Э. Иверсоном [18].

Основное требование к символическому языку, чтобы он был компактным и легко применяемым, чётко описал заданный алгоритм и принцип работы, дал возможность для симуляции т. е. для контроля алгоритма.

Сводное описание самых известных символических языков DDL, Lotis и Iverson находится в [4].

Язык Iverson обладает очень богатой семантикой, но плохо то, что программы не работают. При разработке машинных решений выбрано два направления.

Язык APL [3], [4] является машинным языком, разработанным для ИБМ-360. Это — общий язык для решения проблем, который — благодаря многосторонним операторам и широким возможностям обращения с блоками, — весьма пригодна для симуляции на ЭВМ. Подобно языку Iverson, APL обладает очень богатой семантикой, при его создании сначала была определена семантика, а потом — синтаксис.

ОСС-2 является целевым языком для симуляции электронно-вычислительных систем. Это и есть оффициальный язык для формального описания операционных принципов в Единой Системе Вычислительных Машин [11]. Транслятор разработан в СССР по системе ICL System 4/50 [20], [27]. В Венгрии его адаптация произошла на машине Сименс 4004 Института Координации по Вычислительной Технике в рамках советско—венгерского сотрудничества [6].

В дальнейшем принцип микропрограммного управления будет показан с помощью программ, написанных на языках APL и ОСС-2.

В этом описании система связана с прогрессором только схематично. Директива END «конец исходной программы» принимается во внимание сигнальным разрядом идентификатора END.

Микроадрес '0000' называется начальным адресом (S_k). Под его влиянием начальный адрес следующей подпрограммы в микрокомандах берется

```
APL
▽ WILKES
[1] K ← 2 ⊥ R2 ← R1
[2] US ← C[I ← K + (K = 16) ∧ (1 = TT2 ← □);]
[3] R1 ← S[I + (K > 5) ∨ (K = 5) ∧ (1 = TT1 ← □) ]
[4] → 1 + 4x∧/R1 = SK
[5] → 6 + 1 = END ← □
[6] → ⍴⍴R1 ← OP ← □
[7] ▽
```

```
OCC—2
MODULE WILKES
```

0			F: R1?4!; R2?4!; TT1; TT2; C?16!; S?4!; END; U?16!; C1?16! C2?16!;----; C16A?16!; C1 6B?16!; S1?4!; S2?4!; ----; S5A?4!; S5B?4!; ---- S17?4!
1			R2: = R1
2	1	(R2 = 4)	GT3
	2	≠(R2 = 4)	GT5
3	1	TT1 = 1	C: = C5; S: = S5A
	2	TT1 = 0	C: = C5; S: = S5B
4			GT9
5	1	(R2 = 17)	GT6
	2	≠ (R2 = 17)	GT8
6	1	TT2 = 1	C: = C1 6A; S: = S16
	2	TT2 = 0	C: = C1 6B; S: = S17
7			GT9
8	1	R2 = 0	C: = C1; S: = S1
	2	R2 = 1	C: = C2; S: = S2
	⋮	⋮	⋮
	14	R2 = 16	C: = C15; S: = S15
9	1	(S = 0)	GT12
	2	≠(S = 0)	GT10
10			R1: = S
11			GT1
12	1	END = 1	GT15
	2	END = 0	GT13
13			R1: = U?0/3!
14			GT1
15			OUT
E			

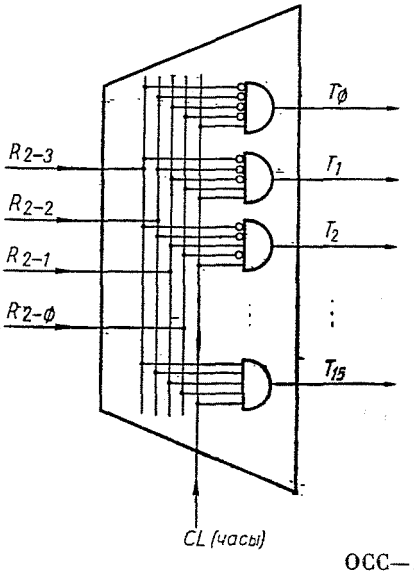
Рис. 4. Моделирование блока управления

не из устройства хранения микропрограмм, а из кода операции команды (Рис. 4).

С целью иллюстрации мощности языка OCC-2 и его сравнения с языком APL показываем техническую симуляцию следующих частей гипотетического устройства управления:

— дешифратор (Рис. 5);
— переключатель (Рис. 6);
— блок регистров (Рис. 7).

В микропрограммной системе длина микрокоманды имеет важное значение. В зависимости от длины слова, т. е. от количества информации, располагаемой в одном слове, различается два типа микропрограммирования:



APL

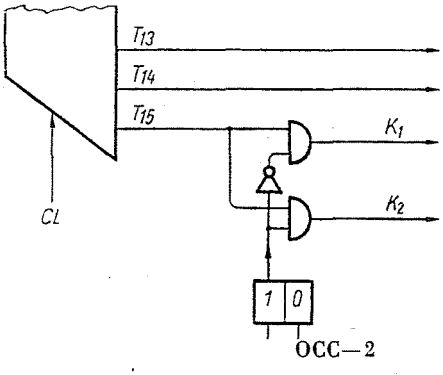
```
[2] T ← (ι15)ε2⊥R2
```

OCC—2

```
 1| TO: = ≠ R2?O !&≠R2?2 !&≠R2?3 !&CL
 2| T1: =   R2?O !&≠R2?2 !&≠R2?3 !&CL

16| T15: = R2?O !& R2?1 !& R2?2 !& R2?3 !&CL
17| C: = TO'MUL'C1 + T1'MUL'C2+ . . . T15'MUL'C16
```

Рис. 5. Аппаратурное моделирование дешифратора



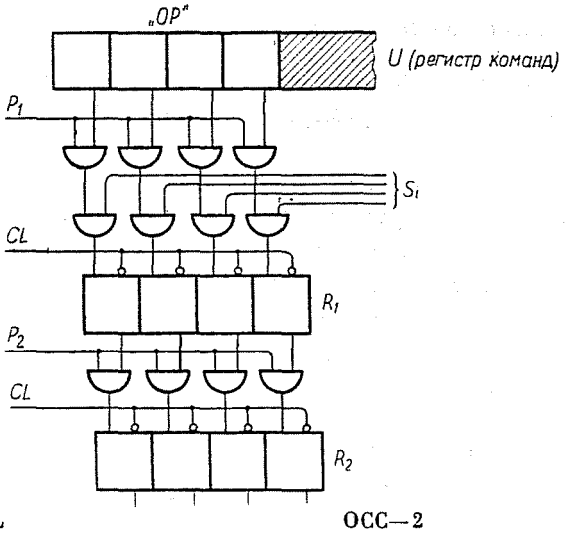
APL

```
[3] K1 ← T [15]∧ ~TT2
[4] K2 ← T[15]∧ TT2
```

OCC—2

```
17| |K1: = T?15 !&≠TT2|
18| |K2: = T?15 !& TT2|
```

Рис. 6. Аппаратурное моделирование переключателя



APL OCC—2

```
[1] R1 ← SI ∨ U[ι4] ∧ (ι4)\P1   |13| |R1: = P1, (4)&U?O/3 !%SI |
[2] R2 ← R1 ∧ (ι4)\P2           |  | |R2: = P2, (4)&R1          |
```

Рис. 7. Аппаратурное моделирование блока регистров

1. *Горизонтальное микропрограммирование*

В целях управления с помощью одной команды всё большим количеством блоков машины команда разделяется на несколько подзон, и каждая подзона управляет одним из технических блоков. В случае сложной системы это приводит к очень длинной команде. Например, в модели ИБМ 360/50 команда занимает 90 разрядов, а в модели 360/65 — 100 разрядов. Работа идёт очень быстро, поскольку блоки, управляемые одним словом, работают одновременно. Но этот метод становится безуспешным, если на известном шаге приходится управлять лишь одним блоком. В этом случае остальные подзоны команды «пустые», нулевые, таким образом часть емкости запоминающего устройства не используется. Другой недостаток при этом, что составлять программы в микрокомандах довольно трудно. В [26] авторы показывают один специальный высокоразвитый язык для составления горизонтальных микропрограмм. Преимущество языка заключается в том, что алгоритмы микроопераций описываются с помощью чрезвычайно простого синтаксиса. Транслятор системы генерирует микропрограмму в табличной форме. На основе таблицы вставление микропрограммы совершается почти непосредственно.

2. *Вертикальное микропрограммирование*

При этом микропрограммы гораздо короче, чем при горизонтальном микропрограммировании. Например, в системе ИБМ Систем 360/25 и Систем 360/75 длина слова равняется 16 разрядам. При вертикальном микропрограммировании команда разделяется на группы. Каждая группа имеет определенное значение — одна группа содержит код микрооперации, а другая — либо адрес операнда, либо изменение операции. При этом то, что при горизонтальном микропрограммировании достигается одной командой, здесь достигается выполнением нескольких команд, зато нет неиспользуемых, нулевых операций. Следовательно, время выполнения команд короче в случае горизонтального микропрограммирования.

Устройство хранения микропрограмм в машине ЕС 1010 тоже вертикального построения (Рис. 8). Это дало возможность отделения друг от друга микрокоманды и микродиректив (executiv) [12]. Для случая вертикального микропрограммирования связь между макрокомандами, микрокомандами и микродирективами показана на рисунке 9.

При выполнении одной машинной команды, необходимо решить следующие задачи:

— изменить содержимое счетчика команд;
— вычислить адрес операнда;
— проделать нужные операции над операндом.

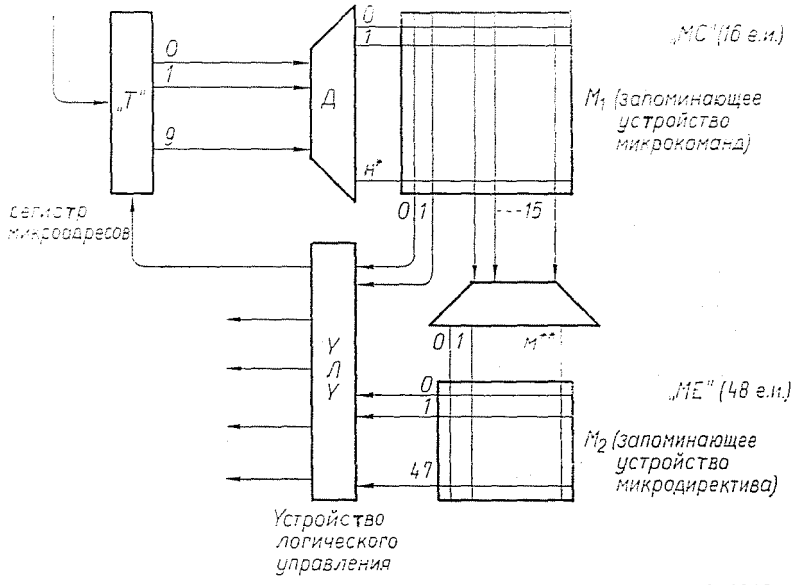


Рис. 8/а. Запоминающее устройство на микропрограммах в машине «ЕС 1010»
* В машине: н = макс 1343
В моделе: н = 5
** В машине: м = 95
В моделе: м = 78

APL

```
∇ EC
[1] MC ← M1 [2⊥T;]
[2] → 3 + (2⊥MC[2 + ι 3]) = 0
[3] → 5 + (2⊥MC[2 + ι 3]) < 7
[4] → 8xρρF ← MC[2 + ι 8]
[5] → 8xρρF ← MC[2 + ι 5], MC[11 + ι 3]
[6] F[ ι 5] ← MC[2 + ι 5]
[7] F[5 + ι 3] ← 0
[8] ME ← M2[(+/(ι 79)x(2⊥F) ε (79ρ256) ⊤ 2⊥, M2[; ι8]); (8 + ι48)]
[9] ∇
```

OCC—2

MODULE EC

0			F: MCO ?26 !; MC1 ?26 !; . . .; MC5 ?26 !; MC?16 !; T?10 !; F?8 !; ME?48 !	
1	1	T = MC0 ?0/9 !	MC: = MC0 ?10/25 !	
	2	T = MC1 ?0/9 !	MC: = MC1 ?10/25 !	
	5	T = MC5 ?0/9 !	MC: = MC5 ?10/25 !	
2	1	MC?2/4 ! = 0	F: = MC?2/9 !	FIESS
	2	(MC?2/4 ! < 0)& (MC?2/4 !7) <	F?0/4 !: = MC?2/6 !; F?5/7 !: = 0	FI
	3	MC?2/4 ! = 7	F: = MC?2/6 !, MC?11/13 !	FIER
3	1	F = 1	ME: = 5320001000000022	HLT
	2	F = 7	ME: = 5320001010000000	TI
	3	F = 10	ME: = 4512743240000000	P + 2P
	79	F = 377	ME: = 4710001000000000	R'UU
4			OUT	
E				

Рис. 8/б. Симуляционные программы

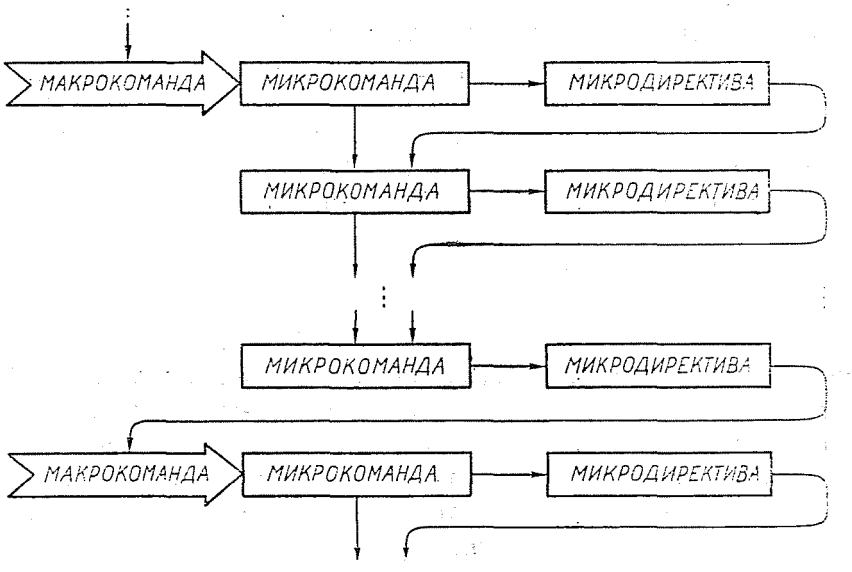


Рис. 9. Микрокоманды и микродирективы в машине «ЕС 1010»

Управление этими операциями в ЭВМ 1010Б осуществлялось в машинном цикле на основе триггеров. Таким образом, выполнение одной довольно простой команды происходило в зависимости от способа адресации за 14—18 циклов, что соответствует 7—9 микросекундам. В машине ЕС 1010 последовательность операций, управляемая одним машинным циклом, выполняется последовательностью операций, начинающихся под влиянием микрокоманды. Таким образом выполнение вышеупомянутой машинной команды (макрокоманды) в машине ЕС 1010 требует 6—8 микрокоманд в зависимости от способа адресации, что соответствует 1,8—2,4 микросекундам. Значит, временные характеристики микропрограммного построения гораздо лучше даже в вертикальном случае.

Конструкция и симуляция

Задача конструктора состоит в создании такой системы программ, которая обеспечивает возможность проектирования ЭВМ на основе схем или формального описания [11]. Подобную систему программ можно составить следующим образом:

1. Исследование структур, обладающих единым и ясным построением системы.

2. Отыскание таких систематических методов, с помощью которых можно создать системы, описанные в пункте 1.

3. Алгоритмизация систематических методов, осуществление проектирования на ЭВМ.

Значит, первой проблемой является исследование структур. Предъявленным требованиям в первую очередь соответствует построение микропрограммированных устройств! В дальнейшем перед конструктором ставится задача составления таких алгоритмов, с помощью которых микропрограммированное устройство умеет изменять содержимое своего устройства хранения для микропрограмм — с целью выполнения наперёд заданных условий работы. Собственно, здесь речь идёт о методах, применяемых ныне в самых современных, так называемых эмуляционных ЭВМ [17]. Из-за широкого применения микропрограмм техническое построение этих машин становится всё более унифицированным. В ходе проектирования больших систем и микропрограмм использовались преимущественно эвристические методы [1]. Известны разные алгоритмы оптимизации электронно-вычислительных систем, устройства хранения микропрограмм [13] и самих микропрограмм [21]. При совершенствовании алгоритмов нет переломного шага в сторону синтеза. Таким образом, нельзя говорить об автоматизированном проектировании, только о проектировании с помощью ЭВМ.

В результате развития технологии создания устройств хранения микропрограмм:

— составляются всё более быстрые, дешевые и мощные микропрограммы;

— во время работы легче осуществляются изменения содержимого устройства хранения микропрограмм.

Естетсвенно, что этими возможностями можно воспользоваться наилучшим образом в том случае, если средствами технического обеспечения ЭВМ будут решаться только задачи чтения или перезаписи из устройства хранения микропрограмм, а также простые задачи установления сигналов. Упрощенно, микропрограмма должна включать в себя следующие:

— микрокоманды по выполнению основной работы (традиционное микропрограммирование),

— специальную микропрограммную часть: под действием алгоритма, выполненного на основе данной микропрограммы, переписываемое поле запоминающего устройства микропрограмм перепишется в машине средствами тесничского обеспечения с целью выполнения наперёд задаваемых рабочих условий.

Одна из причин актуальности микропрограммирования заключается в развитии ассортимента деталей машин. Сегодня сравнительно дёшево можно приобрести мощные устройства хранения. Фирма ТЕКСАС выпускает запоминающие устройства типа „**read only memoria**" (**ROM**) и в устройство заносится информация, указанная заказчиком (интегральные цепочки типа SN 7488A в 256 разрядов и типа SN 74187 в 1024 разрядов). Другие фирмы, изготовляющие полупроводниковые прбиборы, например **National Semiconductor**, выпускают запоминающие устройства типа **ROM**, программируемые покупателем. К этим устройствам можно приобрести и необходимые программирующие приборы. (Это обычно только один раз программируемые запоминающие устройства.)

Ассортимент мощных запоминающих устройств разного построения типа **ROS** расширяется буквально изо дня в день, а их мощность постоянно увеличивается (при снижении удельной цены одного разряда!).

При проектировании микропрограммных устройств управления системы программ, составляемых путём приёмов классического логического проектирования, постепенно заменяются методами, разработанными для высокоразвитых символических языков.

При исследовании методов проектирования безусловно необходима и проверка этих методов. В процессе производства и развития электронновычислительных средств самой трудоемкой и длинной задачей является засечка и контроль. Вопрос чрезвычайно важный, потому что приобретение мощных переписываемых запоминающих устройств микропрограмм в отечественных условиях требует больших затрат. (Применение несколько раз переписываемых запоминающих устройств экономично только для эмуляционных целей.) Решение получается очень просто: необходимо реализовать переписываемое запоминающее устройство симуляций на ЭВМ, а ROM

нужно заказать на основе уже проверенных программ от производящей фирмы.

В ходе производства микропрограммных устройств управления симуляция в двух фазах выступает на первый план.

1. *Фаза проектирования*

При проектировании приходится проверять микропрограмму. В этом случае симуляция играет роль «дополняемой» модели, и на основе результатов можно изменить программу (итерационный метод). Особенность имитационной модели состоит в том, что необходимо имитировать с разной степенью глубины электронную окружающую среду устройства управления (например, оперативную память, арифметическое и логическое устройство и т. д....), поскольку в момент проектирования техническая реализация ещё неизвестна.

В [14] показана симуляционная программа. Для программы характерны высокоразвитый символический язык (ОСС-2) и соответственно этому — большая память.

2. *Фаза засечки*

В процессе засечки необходимо проверить работу отдельных частей устройства, а также их совместную работу. Практически эта задача решается посредством целевого устройства. При этом запоминающее устройство микропрограмм реализуется путём симуляции на ЭВМ в режиме оплайн [2]. С целью контроля выходы готового устройства и его отдельных частей подключаются на входы секвенциальной или комбинированной сети, обладающей некоторым количеством входов и выходов. Наборы сигналов, поступающих на входы сети, в дальнейшем будут называться управлениями, а наборы сигналов на выходах сети — ответами.

Имеем два способа осуществления контроля:

а) Генерируем все возможные последовательности управлений, которые только могут поступать с контактных шин присоединённых к машине устройств. На основе полученных ответов устанавливаем, выполнились ли предписанные работы под влиянием генерированных последовательностей импульсов.

б) Зная структуру проверяемого устройства, определяем такую последовательность управлений, которая проверяет каждую структурную часть машины, без генерирования всех возможных комбинаций импульсов. Условие проверяемости, чтобы под действием каждого управляющего сигнала происходили изменения, за которыми можно наблюдать имеющимися средствами, или если происходят быстрые асинхронные процессы, то можно будет проверять правильность их прохождения дополнительно. Необходимо создать

такую симуляционную систему, которая обеспечивает возможность обмена информацией между ЭВМ и любым электронно-вычислительным устройством. То есть с помощью симуляционной системы ЭВМ умеет имитировать любое электронно-вычислительное устройство — в данном случае запоминающее устройство микропрограмм, — работающее по наперёд заданным условиям.

Симуляционная система состоит из двух частей:

— из специальной периферии, присоединяющейся к вычислительной машине,

— из системы программ.

С одной стороны симулятор подключается на входной и выходной каналы вычислительной машины. А с другой стороны симулятор устроен так, что он и ЭВМ совместно имитируют запоминающее устройство микропрограмм. Если к этой стороне симулятора подключить остальные блоки процессора, то на основе заранее введённой микропрограммы ЭВМ совершает симуляцию запоминающего устройства микропрограмм и функционально проверяет польный собранный механизм.

Новейшие достижения в области исследования микропрограммных управлений

В области проектирования вычислительных машин появился очень надёжный метод, — метод применения так называемых блоков клеточной структуры. (Programmable Cellular Arrays — PCA/ [19]. Вопросы технологического осуществления здесь не рассматриваются. Структура блоков похожа на двухмерную решётку, где на перекрестках помещены совершенно одинаковые цепи тока. Каждую клетку, цепь тока, расположенную на перекрестках, можно запрограммировать в 12 различных рабочих состояний. Состояние клеток, то есть программу системы можно изменять многократно как угодно. Соответствующим выбором состояний можно реализовать различные микропрограммы. В каждой клетке находится несколько запоминающих элементов, которые в свою очередь определяют программы, записанные в PCA. Состояние отдельных клеток зависит от информации, содержащейся в этих запоминающих элементах. Кроме этого в каждой клетке есть такое запоминающее устройство, состояние которого может быть изменено выполняющей программой. Эта возможность означает, что программа может использовать эти запоминающие устройства в качестве разрядов одного регистра. Таким образом дополнительные части процессора (регистры, комбинационные сети) реализуются тоже в этом блоке. Алгоритм работы должен быть внесён в ЭВМ. Единственный критерий при задании алгоритма, чтобы можно было пользоваться только определенным набором элементов. Сначала выбираются те части алгоритма, которые могут быть выполнены с помощью микрокоманд. На основе этих частей можно определить состояние строк PCA.

(Одна микрокоманда, т. е. одна микрооперация соответствует одной строке матрицы.) Затем по алгоритму нужно обеспечить секвенцию строк. Для решения этой задачи используются остальные запоминающие элементы в строках матрицы.

Задачи, решаемые ЭВМ при проектировании РСА:

1. Определение состояния клеток, реализующих микрокоманды.

2. Определение состояния клеток, обеспечивающих секвенцию (последовательное выполнение) команд.

Программируемые блоки клеточной структуры применяются в больших системах типично для решения таких специальных задач, как, например, выполнение арифметических операций. Применяемость в большой степени увеличивается за счет того, что клетки можно перепрограммировать. Таким образом, тот же самый блок в разные моменты времени может выполнять разные функции по оптимальному алгоритму решаемой задачи. Использование здесь кратко изложенного РСА представило возможность для разработки методов микропрограммного проектирования на основе схем. Конечно, этот метод основан на особенностях РСА, и в случае других структур придется пользоваться другими методами.

Вторая возможность для перепрограммирования — это упомянутая ранее эмуляция. Записав в такую эмуляционную систему микропрограмму любой другой системы, выполняется работа последней [16].

По нашему мнению критерием дальнейшего развития в сторону синтеза является то, чтобы вместо эвристических методов исследования проводились на глубокой теоретической основе. В области интегрированных систем необходимо придавать большую роль теории автоматики и теории графов [23]. Собственно, в рамках разработки микропрограммной диагностики необходимо распространить услуги, применяемые для потребительских и системных программ (это привычно в случае макропрограмм), и на систему микропрограмм [25]. Вместо микроязыков типа ассемблер [17] целесообразно применять высокоразвитые языки, которые сосредотачиваются на преобразования в регистрах [30]. В случае применения APL очень удобно, что микропрограмма для устройства управления, управляемого объекта и алгоритма управления пишется на одном и том же языке. Сложная система становится более гибкой за счет многосторонней применяемости. Предлагаем внутри системы APL определение целевых программных подмножеств (Subset).

При подключении электронных приборов, обладающих двумя устойчивыми (1/0), из-за большой гибкости преимущество придаётся микропрограммному включению [9], [31].

Интересная тенденция — в первую очередь в случае вычислительных машин, — что микропрограммирование берёт на себя большую часть программирования на машинном языке. В виде микропрограмм составляются ессемблеры, симуляторы, лоадеры (программы ввода), трансляторы и другие

простые операционные системы [22], [34]. Относительно скорости работы это приводит к 10 кратному увеличению.

В трансляторах высоких языков микропрограммы используются и непосредственно, так например, микропрограммами можно выполнять анализ текстов [28] или проверку синтаксиса [8]. В [15] автор использовал микропрограммы для составления системы APL. Естественно, что новые решения появляются и в организации главной памяти, т. е. оперативного запоминающего устройства [23], [33].

Микропроцессоры, применяющиеся в чрезвычайно широких областях, также проектируются с микропрограммным управлением [29]. Таким образом, применение цепей высокой недёжности LSI (Large-Scale Integration) становится возможным.

Резюме

Микропрограммное управление относится к важнейшим областям вычислительной техники. Суть микропрограммного управления состоит в том, что для осуществления управления машиной задаётся единая, чёткая система, что в свою очередь создаёт возможность машинного проектирования и моделирования. При реализации применяются такие схемы и элементы цепей тока, производство которых началось именно в последние годы. Здесь имеются в виду запоминающие устройства большой скорости и мощности типа LSI. Без изменения основных элементов схемы можно построить чрезвычайно гибкие, так называемые эмуляционные системы.

В этой статье значение высоких символических языков подчёркивалось неоднократными средствами проектирования.

Мы поставили перед собой цель, дать общие сведения о микропрограммировании и дать представление о методах симуляции на ЭВМ. Параллельно с этим мы всё подчеркиваем значение технологии, так как новые структуры и алгоритмы оказываются недействительными без соответствующего технологического обеспечения.

Литература

1. A. M. Abd-Alla, D. C. Karlgaard: Heuristic Synthesis of Microprogrammed Computer Architecture IEEE Trans. on Computers Vol C-23. No. 8. Aug. 1974. pp. 802—808.
2. Arató P.—Kalmár P.—Kondorosi K.—Lantos B.: Digitális berendezések számítógépes szimulációjának és on-line beмérésének egy módszere, Programozási rendszerek '72 Konferencia, Szeged, 1972.
3. P. C. Berry: APL 360 Primer, IBM Corporation, 1968.
4. Bohus M.—Flesch I.—Géher K.—Németh G.—Pápay Zs.—Szittya O.—Theisz P.: Logikai rendszerek számítógépes szimulációja. Tanulmány a Számítástechnikai Koordinációs Intézet számára 1969.
5. Bohus M.: Gépi tervezési eljárás digitális rendszerek strukturális és logikai szintézisére. Számítógéptechnika 74' Konferencia 1974. Esztergom.
6. Bohus M.: OSzSz Szimulációs programnyelv. Szovjet—magyar együttműködés digitális rendszerek szimulációs problémáinak megoldására. Mérés és Automatika XXII. évf. 1974. No. 12.
7. Bohus M.: Szimbolikus nyelvek felhasználása digitális rendszerek funkcionális és parametrikus szimulációjára. Híradástechnika XXIII. évf. 6. sz.
8. Y. Chu: Recursive microprogramming in a Syntax recognizer Micro 6, Preprints, Sept. 1973. pp. 91—98.
9. D. Dromard, O. Gibergues: A microprogrammed data communications procedure controller in Micro 6, Preprints, Sept. 1973. pp. 76—79.

10. J. R. Duley, D. L. Dietmeyer: A Digital System Design Language (DDL) IEEE Trans. on Computers Vol. C-17. Sept. 1968. pp. 850—861.
11. Единая Система Электронных Вычислительных Машин формальное Описание принципов операций ОСТ: 50.000.010 МОСКВА 1970.
12. ESz 2010 Központi egység. Működési leírás VIDEOTON 270 10020 01 0/A
13. A. Grasseli, U. Montanari: On the minimization of READ-ONLY memories in microprogrammed digital computers. IEEE Trans. on Comp. Vol C-19 pp. 1111—1124 Nov. 1970.
14. Hantos P: Mikroprogramozott vezérlések szimulációja Diplomaterv 1973. BME.
15. A. Hassitt, J. W. Lageschulte, L. E. Lyon: Implementation of a high level language machine. Commun. Assoc. Comp. Mach. Vol 16. Apr. 1973. pp. 199—212.
16. L. W. Hoevel: "Ideal" Directly Executed Languages: An Analytical Argument for Emulation IEEE Trans. on Comp. Vol C-23 No. 8. Aug. 1974. pp. 759—768.
17. S. S. Husson: Microprogramming Principles and Practices, Prentice-Hall, Inc. Englewood Cliffs, N. Y. 1970.
18. K. E. Iverson: A Programming Language, John Wiley, 1962.
19. J. R. Jump, D. R. Fritsche: Microprogrammed Arrays IEEE Trans. on Computers 1972. Szept. 975—984.
20. Язык для описания структурных алгоритмов и схем (ОСС-2) ОСТ: 4.010.000.025 МОСКВА 1970.
21. R. L. Kleir, C. V. Ramamoorthy: Optimization Strategies for microprograms. IEEE Trans. on Computer Vol C-20, pp. 783—794. July 1971.
22. Microprogramming Guide for the Hewlett-Packard 2100 Computer. Hewlett-Packard Co. Doc. 5951—3028. Feb. 1972.
23. H. D. Mills: Mathematical foundations for structured programming. IBM Corp. Rep. FSC 72—6012. Feb. 1972.
24. S. Pakin: APL 360 Reference Manual. Science Research Associates, 1968.
25. C. V. Ramamoorthy, L. C. Chang: System modeling and testing procedures for microdiagnostics. IEEE Trans. on Comp. Vol C-21. pp. 1169—1183. Nov. 1972.
26. C. V. Ramamoorthy, M. Tsuchiya: A High-Level Language for Horizontal Microprogramming. IEEE Trans. on Comp. Vol-C-23. No. 8. Aug. 1974. pp. 791—802.
27. Руководство по использованию языка ОСС-2 (Научно-Исследовательного центра Электронно-Вычислительной Техники НИСЭВТ) МОСКВА 1971.
28. P. S. Roberts, C. S. Wallace: A microprogrammed lexical processor Proc. IFIPS 1971. London: North-Holland 1972. pp. 577—581.
29. G. W. Schultz, R. M. Holt, H. L. McFarland, Jr.: A guide to using LSI microprocessors COMPUTER Vol. 6. 5 pp. 13—19. June 1973.
30. SIGPLAN/SIGMICRO Interface Meeting Preprints, May 1973.
31. F. D. Strout: Microprogramming in the hierarchy of peripheral control. IEEE Int. Comp. Soc. Conf. 1971. pp. 111—112.
32. R. T. Thomas: Computer organization for allowing dynamic user microprograming. Assoc. Comput. Mach. SIGMICRO News-letter 4,2, July 1973. pp. 28—42.
33. R. T. Thomas: Organization for Execution of User Microprograms from Main Memory: Synthesis and Analysis. IEEE Trans. on Comp. Vol-C-23. No. 8. Aug. 1974. pp. 783—791.
34. A. H. Werkheiser: Microprogrammed Operating Systems Proc. 3. Annu. Workshop Microprogramming Oct. 1970.
35. M. V. Wilkes: The Best Way to Design an Automatic Calculating Machine, Report of Manchester University Computer Inaugural Conference, Manchester, 1951. July.
36. M. V. Wilkes, W. Renwick, D. Weeler: The Design of a Control Unit of an Electronic Digital Computer, Proc. IEEE 105 (1958) 121.

Петер Хантош
научный стипендиант
Будапештский Технический Университет
H-1521