最新技術

Microprogram과 컴퓨터의 制禦裝置

趙 廷 完
韓國科學院 電算室長 · 工博

1. 序 論

지난 一世紀동안에 컴퓨터 하드웨어 分野는 特記할만한 發展을 하였다. 그 例로는 集積回路와 半導體 기억장치의 活用으로 이것은 主로 半導體 처리技術의 發達에 기인하는 것이다. 이보다 더 획기적인 事實은 半導體 기억장치의 使用技術의 開發과 컴퓨터의 中央演算裝置와 같은 대규모의 random logic의 集積 즉 LSI技術의 發展이다. 그러나 1950年代 初에 發表된 microprogram이 없었더라면 半導體 기억장치와 random logic LSI가 產在와 같이 광범위하게 活用될 수는 없었을 것이다. Microprogram이 아니었더라면 같은 프로그램을 여러개의 다른 모형의 컴퓨터에 使用할 수 있는 IBM S/360와 같은 컴퓨터의 개발이 不可能하였고 現在의 microprocessor와 같이 中央演算裝置를 한개의 chip에 넣을 수 있는 技術이 있어도 이것을 만들 必要는 없었을 것이다.

Microprogram은 主로 컴퓨터의 制禦裝置에 많이 使用되며 이를 爲한 制禦用 기억장치(Control Store; CS)를 별도로 둔다. 이러한 경우 보통 컴퓨터의 電子論理回路로 된 制禦裝置를 CS에 기억된 microprogram으로 代置한다. 電子回路를 프로그램으로 代置한다는 것은 컴퓨터의 設計過程과 改善過程에 重大한 의의가 있다. 복잡한 回路設計問題는 프로그램設計문제로 되며 따라서 經濟的인 面과 時間的인 面에서 利益을 얻을 수 있다. 그러나 microprogram의 경우 보통의 프로그램과 마찬가지로 電子回路보다 그 수행시간이 느리므로 대단히 빠른CS를 使用하지 않는 限 速度 때문에 그 使用이 實用的이 아니다. 이러한 理由때문에 1950年代 初에 發表된 microprogram이 半導體 기억장치가 發達된 最近에 와서야 活用되게 되었다. 近年에 와서는 microprogram으로 된 制禦裝置를 가진 microprogrammed 컴퓨터는 물론 심지어 使用者가 직접 원하는 instruction set를 만들어 使用할 수 있도록 제작된 microprogrammable 컴퓨터도 많이 제작되고 있다.

2. Microprogram

Microprogram은 制禦裝置를 設計하고 實現하는 方法中 하나이다. 우선 컴퓨터의 制禦裝置에 關하여 論한다. 컴퓨터의 制禦裝置는 프로그램 수행에 必要한 信號를 clock pulse에 맞추어 發生하는 裝置이다. 따라서 制禦裝置는 instruction fetch, decoding, instruction의 데이타 部分을 利用한 演算에 必要한 데이타 回線의 gating, 그

리고 컴퓨터가 다음 instruction을 수행할 수 있도록 컴퓨터의 상태를 바꾸어 주는 일을 하여야 한다. 換言하면 주기억장치에 기억된 一般的인 프로그램은 그 instruction 하나 하나가 컴퓨터에게 무엇을 하라고 指示하는 것인 反面에 制禦裝置는 컴퓨터가 各 instruction을 수행하기 爲하여 컴퓨터의 어느 部分을 언제 어떻게 利用할 것인가를 指示하는 것이다.

一般的인 制禦裝置, 즉 論理回路로 만든 制禦裝置는 보통 여러개의 decoder와 f/f으로 되어 있다. 一般的으로 制禦裝置의 論理回路는 대단히 복잡하여 古典的인 方法으로는 그 設計가 거의 不可能하여 體系的인 設計를 할 수 없으므로 設計비용과 時間이 많이 소용된다. 그 一例로 8-bit op-code를 갖는 컴퓨터의 경우 256개의 다른 op-code를 가질 수 있으므로 8개의 Level 입력과 수개의 clock 입력을 가지며 256개의 출력을 갖는 sequential 論理回路를 設計하여야 되므로 그 복잡성을 짐작할 수 있으리라 믿는다. 그림 1은 論理回路로 만든 制禦裝置와 그 周邊部分을 나타낸 것이다.

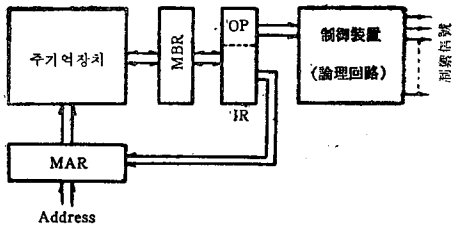

그림 1 論理回路로 만든 制禦裝置

그림 1의 動作을 간단히 說明하면 주기억장치에 들어있는 프로그램의 instruction이 MAR(memory address register)에서 指定하는 住所에서 BBR (memory buffer register) 그리고 IR(instruction register)로 옮겨지면 IR의 op-code部分이 制禦裝置로 보내져서 decoding 過程을 거쳐 clock pulse에 맞추어 op-code의 bit 형태에 따라 다른 制禦信號를 發生한다.

Microprogram으로 制禦裝置를 만들 경우 우선 그 設計가 간단하다. 아무리 制禦部分이 복잡하더라도 microprogram의 경우 回路設計 대신 program 작성을 하면 되며 더군다나 各 op-code에 對한 設計를 獨立的으로 할 수 있다. 論理回路의 경우 各 op-code에 對한 獨立的인 設計는 비용관계로 實質的이 아니다. 앞의 例의 경우 256개의 instruction을 갖고자 하면 우선 256개의 microprogram을 作成하면 되며 만일 중복되는 部分을 회피하고자 하면 적당한 conditional transfer 等에 依하여 實現할 수 있다. 그림2는 microprogram에 依한 制禦裝置와 그주변 部分을 나타낸 것이다. 물론 모든 microprogram에 依한 制禦裝置를 갖는 컴퓨터가 그림 2와 같

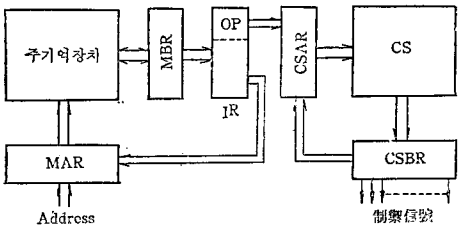

그림 2 Microprogram으로된 制禦裝置

은 것은 아니며 여기서는 개념적인 형태를 소개한 것이다. 그림2의 動作을 간단히 說明하면 주기억장치에 들어있는 프로그램의 instruction이 MAR에서 指定하는 住所에서 MBR 그리고 IR로 옮겨지면 IR의 op-code 部分이 CSAR(control store address register)로 옮겨져서 CS의 住所를 指定하게 되고 그 住所의 內容이 CSBR (control store buffer register)로 들어와 CSBR의 bit들이 바로 制禦信號로 clock pulse에 時間을 맞추어 制禦한다. CS의 한개의 word는 micro-instruction이라 하며 한개의 op-code

를 수행하기 爲하여는 한개 以上의 micro-instruction이 必要할 수도 있으며 이를 microprogram이라 한다. 또 한개의 micro-instruction은 여러개의 micro-operation으로 구성될 수 있는데 micro-operation이라 함은 한개의 register 內容이 다른 register를 옮겨지는 것을 意味하며 따라서 한개의 micro-instruction은 여러개의 micro-operation을 포함할 경우도 있으며 이때는 multiphase clock에 동기시켜 수행하도록 한다. 또 microprogram의 경우 一般的인 프로그램 수행과 달리 별도로 PC(program address register)를 두지 않는 것이 상례이며 따라서 다음 micro-instruction의 住所는 보통 現在 fetch된 micro-instruction에서 직접 혹은 간접적으로 表示한다. 그림2에서 CSBR에서 CSAR로 향하는 화살표는 이것을 의미한다. microprogram에 依한 制禦體系를 만들 경우 以上에서 論한 것으로 짐작할 수 있드시 microprogrammer는 論理回路로 制禦裝置를 設計할 때와 마찬가지로 모든 micro-operation과, 모든 데이타의 回線, gate, 그리고 컴퓨터에서 並行으로 일어날 수 있는 모든 動作을 철저히 알고 있어야 한다.

Microprogram에 依한 制禦裝置를 채택할 경우 하나의 기계어의 instruction을 수행하기 爲하여 그 op-code에 해당하는 microprogram을 수행하여야 하는데 그러기 爲하여는 micro-instruction을 CS로 부터 fetch하고 clock pulse의 적당한 위상에 따라 制禦信號를 發生하여야 한다. 그러므로 論理回路에 依한 制禦裝置보다 instruction 수행에 時間이 많이 소용된다. 따라서 CS는 주기억장치보다 그 cycle time이 훨씬 빠른 것을 使用하여야 한다. 따라서 CS의 우선 조건은 non-destructive read-out이어야 한다. 더우기 CS는 쉽게 변형할 수 없는 기억장치이어야 된다. 그렇지 않으면 고의로 혹은 부주의로 CS의 內容을 변환시키면 정상적인 動作을 기대할 수 없기 때문이다. 따라서 CS는 Rom (read only memory) 혹은 writable Ron으로 만든다.

3. Micro-Instruction Format

Micro-instruction은 一定한 時間, 즉 한개의 machine cycle 동안에 일어날 수 있는 micro-operation들을 始動시키거나 수행시키는 것이다. Micro-operation은 制禦하는 대상에 따라 다르며 한 machine cycle 동안에 하나의 gate를 制禦하는 것으로 부터 adder와 같은 하나의 subsystem까지도 制禦할 수 있다. 이와같이 micro-operation이 制禦하는 대상의 大少에 따라 micro-instruction은 크게 두가지 format으로 나눌 수 있다. 첫째는 水平 혹은 minimally encoded format이며 둘째는 垂直 혹은 maximally encoded format이다.

水平 micro-instruction format은 micro-instruction의 各 field가 直接 데이타 回線을 制禦한다. 따라서 데이타 回線數에 比例하여 microinstruction word의 크기가 커진다. 그림 3은 컴퓨터의 中央演算裝置의 一部分을 나타낸 것이다. 그림3에서 adder의 左上部에 T/C라 함은 가bus의 데이타를 adder를 보낼때 데이타를 그대로 보낼 것인가 혹은 이의 complement를 보낼 것인가를 制禦하기 爲한 것이고 adder의 右便의 +1이란 信號는 1을 合할 것인가 아닌가를 制禦하는 것이다. 또 各 화살표를 한 連結線에 ⓐ, ⓑ, ……, Ⓟ와 같은 表示를 한것은 그 連結線에 데이타가 나타나도록 하는 gate의 制禦信號의 이름을 나타낸다. 例를 들어서 ⓔ에 信號가 있으면 다bus 上의 데이타가 register R1으로 들어

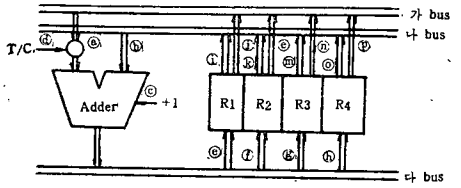

그림 3 간단한 演算裝置

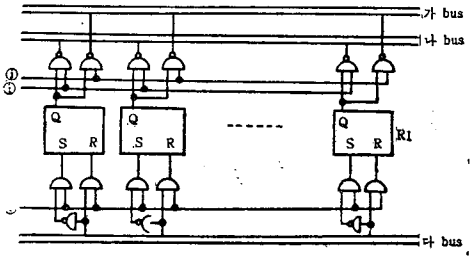

그림 4 Qatiug信號 ⓙⓘ와 ⓙ

감을 의미한다. 이와같은 動作원리는 그림4에서 gating信號 ⓔ와 ⓘ를 例를 들어서 圖示하였다. 그림3과같은 體係에서 만일 register R1과 R2의 데이타를 合하여 그 결과를 R3에 보관하려고 할 때 必要한 microprogram을 水平 micro-instruction format으로 作成한 것은 그림 5와 같다.

住 所	a b c d e f g h i j k l m n o p next	結 果
0000	0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0001	(R1+R2)
0001	0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 —	(→R3)

그림 5 micro-instruction format

그림 5의 microprogram을 관찰하면 첫번째 micro-instruction은 i와 l-field가 1이므로 R1과 R2의 데이타를 各各 나와 가bus에 실리는 micro-operation을 한다. 물론 두개의 動作이 並行한다. 가와 나bus는 各各 adder의 左右 operand로 연결되어 있기 때문에 두개의 數가 adder를 利用하여 合하여지고 計算이 끝나면 그 결과가 다 bus에 실린다. 두번째 micro-instruction은 g-field가 1이므로 다 bus의 데이타가 R3로 들어가서 원하는 計算이 完了된다.

垂直 micro-instruction format은 보통 흔히 볼 수 있는 기계어의 형태와 비슷하다. micro-instruction은 몇개의 field로 나누어져 있어서 各 field의 bit 형태를 직접 制禦信號로 사용하는 것이 아니라 各 field를 decoder를 通하여 制禦信號를 얻는다. 그림3의 演算裝置를 利用하여 앞의 例를 垂直 micro-instruction format로 만든 것은 그림 6과 같다.

Source		Destination	T/C	C	Next	結 果
Left	Right					
R2	R1	R3	0	0	—	R1+R2 →R3

그림 6. 垂直 micro-instruction format

그림 6에서 그림 3에 주어진 演算裝置의 경우 처음 3 field는 2 bit씩이면 되고 나머지 2bit, 모두 合하여 8bit이면 充分하다. 그리고 source와 destination을 나타내는 field들은 decode를 하여야만 必要한 信號를 發生할 수 있다.

水平과 垂直 micro-instruction을 비교하면 水平 micro-instruction은 各 field의 bit를 직접 制禦信號로 使用하기 때문에 micro-instruction의 길이가 그림 5에서 보듯이 길은 반면에 몇개의 動作을 並行할 수 있다. 또 micro-instruction의 field들이 직접 데이타 回線을 制禦하므로 故障診斷 프로그램을 作成하는데 좋은 format이다. 그 反面에 micro-instruction의 길이가 길어서 CS가 많이 必要하고 microprogram作成이 어렵다. 垂直 micro-instruction format은 기계어와 같이 간단명료하여 microprogram 作成하기가 쉬우며 micro-instruction 길이가 짧아서 CS word의 길이가 작아도 된다.

一般的으로 水平 micro-instruction format은 IBM S/360과 같은 大型 microprogrammed 컴

퓨터에 적용되며 垂直 micro-instruction format은 Inter Data 3 혹은 4와 같은 小型 혹은 中型 micro-programmable 컴퓨터에 많이 적용된다.

4. Microprogram의 長短點

本節에서는 microprogram의 論理回路에 對한 相對的인 長短點을 考察한다.

컴퓨터의 設計過程에서 microprogram에 依한 制禦裝置는 論理回路로 만드는 것보다 용이하다. 앞서 論한 것과 같이 制禦裝置는 入出力 變數가 많은 대단히 복잡한 回路이므로 古典的인 Karnaugh map이나 Quine-McCluskey 方法과 같은 體係的인 設計方式을 使用하기 어렵다. 反面에 microprogram에 依한 制禦裝置는 컴퓨터의 中央演算裝置. 入出力裝置와 같은 다른 subsystem으로 부터 獨立的으로 設計가 可能하며 심지어 다른 subsystem의 設計가 完了되었을때 이들에 알맞는 制禦裝置의 設計도 可能하다. Microprogram의 경우 simulator를 利用하여 microprogram의 作成, debugging, checking 등이 용이하다. 또 論理回路로된 制禦裝置의 경우 컴퓨터의 設計過程에서 많이 發生하는 subsystem 혹은 instruction set의 약간의 변형에도 論理回路가 modular 구조가 아니므로 全體的인 再設計가 必要할 경우가 많은 反面에 microprogram의 경우 해당하는 micro-routine만 수정하면 되며 경우에 따라서는 CS를 교환하면 된다.

Microprogrammed 컴퓨터는 또한 instruction set의 융통성을 가질 수 있다. 즉 하나의 컴퓨터 設計를 가지고 instruction set이 다른 여러가지 컴퓨터의 實現이 可能하며 反對로 여러가지로 設計된 컴퓨터에 같은 instruction set를 使用하도록 만들 수 있다. 前者는 writable Rom을 利用한 microprogrammable 컴퓨터를 의미하며 사용자가 自己가 必要한 instruction set를 만든 後 이에 必要한 制禦信號들을 發生할 수 있는 microprogram을 作成하도록 된 컴퓨터이다. Varian 72, Inter Data 3 等이 이러한 컴퓨터이다. 後者는 여러가지 다른 컴퓨터에 microprogram을 利用하여 같은 instruction set를 使用하도록 하는 컴퓨터이다. 이 경우 물론 이들 microprogram은 서로 달라야 한다. 이러한 例는 IBM S/360의 여러 model들과 같은 microprogrammed 컴퓨터와 emulator이다.

Microprogram은 論理回路의 制禦裝置보다 그 Package가 간단하다. 現在의 技術로는 論理回路의 경우 SSI나 MSI 回路의 利用이 可能하나 아직 package面에서 볼때 microprogram을 爲한 Rom과 비교하여 電子素子가 많이 必要하며 따라서 신뢰도가 낮다.

Microprogram의 短點은 論理回路에 比하여 그 수행시간이 느린 것이다. 우선 첫째로 CS의 access time이 문제가 되는 것으로 실제적인 컴퓨터의 cycle time은 CPU (central processing unit)의 cycle time과 CS의 cycle time의 合으로서 현재까지 CPU cycle과 CS access가 並行하지 못하기 때문이다. 수행시간을 줄이는 立場에서는 各 instruction에 해당하는 micro-routine은 되도록 짧아야 CS access數를 줄일 수 있고 또 micro-operation들의 並行이 可能한 水平 micro-instruction format이 利益이나 水平 micro-instruction format는 그 instruction Word의 길이가 길어서 경제적이 아니다. 따라서 大部分의 경우 경제성과 수행시간 그리고 microprogram하기 편리한 立場에서 水平과 垂直을 절충한 micro-instruction format을 택한다.

5. 結 論

Access time이 빠른 半導體 기억장치 技術의 계속적인 發展으로 因하여 microprogram은 컴퓨터의 制禦裝置 設計에 重要한 역할을 한다. 그러나 microprogrammer는 컴퓨터의 論理回路 및 architecture 그리고 timing 等에 關하여 철저한 지식을 갖어야 하므로 microprogrammer의 수효는 一般 프로그래머의 수효와 같이 增加하지 못한다.

Microprogram의 응용은 컴퓨터의 制禦裝置, emulator, algorithmic processor, diagnostic test routine 等의 많은 응용分野가 있으며 앞으로의 硏究는 制禦裝置는 물론 microprogram의 많은 보급을 爲한 higher-level microprogramming language의 compiler, microprogram 使用을 實質的으로 할 수 있도록 CS뿐만 아니라 사용자가 주기억장치에 microprogram을 하여 使用할 수 있는 컴퓨터 architecture 等의 方向이 많으리라 기대된다.

참 고 문 헌

1. M.V. Wilkes, W. Renwick and D. Weeler, "The Design of a Control Unit of an Electronic Digital Computer," Proc. IEEE, 105 (1958), pp. 121.
2. H.T. Glantz, "A Note on Microprogramming, "JACM, 3(1956) pp.77.
3. S.S. Husson, Microprogramming: Principles and Practices, Prentice Hall, Englewood Cliffs, N.J., 1970.
4. J.M. Kurtzberg and R.D. Villani,"A Balanced Pipelining Approach to Multiprocessing on an Instruction Stream Level," IEEE Trans. on Computers, Vol. C-22, No.2, Feb. 1973, pp. 143.
5. C.V. Ramamoorthy and L.C. Chang, "System Modeling and Testing Procedures for Microdiagnostics, "IEEE Trans. on Computers, Vol. C-21, No.11, Nov. 1972, pp. 1169.
6. C.V. Ramamoorthy and M. Tsuchiya, "A High-Level Language for Horizontal Microprogramming," IEEE Trans. on Computers, Vol. C-23, No.8, Aug. 1974, pp. 791.
7. R.T. Thomas, "Organization for Execution of User Microprograms from Main Memory: Synthesis and Analysis," IEEE Trans. on Computers, Vol.C-23, No.8, Aug. 1974, pp. 783.