4장 / 명령 실행과 제어

정보처리산업기사

SECTION 55

마이크로 오퍼레이션
(Micro Operation)

마이크로 오퍼레이션의 정의

- https://www.youtube.com/watch?v=urqPobwPOzs
- 마이크로 오퍼레이션은 Instruction을 수행하기 위해 CPU 내의 레지스터와 플래그가 의미 있는 상태 변환을 하도록 하는 동작이다.
- 마이크로 오퍼레이션은 레지스터에 저장된 데이터에 의해 이루어지는 동작이다.
- 마이크로 오퍼레이션은 한 개의 Clock 펄스 동안 실행되는 기본 동작으로, 모든 마이크로 오퍼레이션은 CPU의 Clock 펄스에 맞춰 실행된다.
- 마이크로 오퍼레이션은 컴퓨터의 모든 명령을 구성하고 있는 몇 가지 종류의 기본 동작으로, 컴퓨터 프로그램에 의한 명령의 수행은 마이크로 오퍼레이션의 수행으로 이루어진다.
- 마이크로 오퍼레이션의 순서를 결정하기 위해 제어장치가 발생하는 신호를 제어 신호라고 한다.
- 마이크로 오퍼레이션은 명령을 수행하기 위해 진행되는 가장 작은 단위의 동작을 의미하는 것으로, 더 이상 세분화될 수 없고 실행 중에는 중단되지 않는다는 의미에서 원자 연산 이라고도 한다.
- 마이크로 오퍼레이션은 Instruction 실행 과정에서 한 단계씩 이루어지는 동작으로, 한 개의 Instruction은 여러 개의 마이크로 오퍼레이션이 동작되어 실행된다.

마이크로 사이클 타임(Micro Cycle Time)

- 한 개의 Micro Operation을 수행하는 데 걸리는 시간을 Micro Cycle Time 이라 한다.
- 모든 순서 논리 회로는 Clock Pulse의 동기화에 의해 동작되는데, CPU도 하나의 거대한 순서 논리 회로 이므로 CPU 역시 이 Clock Pulse에 동기화되어 동작된다. 이 때의 Pulse를 CPU Pulse이라 하며, 한 개의 Micro Operation은 이 CPU Clock의 발생 주기의 간격 시간 내에 실행된다.
- CPU Cycle Time 또는 CPU Clock Time이라고도 하며, CPU 속도를 나타내는 척도로 이용한다.

Micro Cycle Time 부여 방식

- Micro Cycle Time은 CPU 클럭 주기와 Micro Cycle Time의 관계에 따라 동기 고정식, 동기 가변식, 비동기식으로 구분된다.

- 동기 고정식(Synchronous Fixed)
  - 모든 마이크로 오퍼레이션의 동작시간이 같다고 가정하여 CPU Clock의 주기를 Micro Cycle Time과 같도록 정의하는 방식이다.
  - 동기 고정식은 모든 마이크로 오퍼레이션 중에서 동작시간이 가장 긴 마이크로 오퍼레이션의 동작시간을 Micro Cycle Time으로 정한다.
  - 동기 고정식은 모든 마이크로 오퍼레이션의 동작시간이 비슷할 때 유리한 방식이다.
  - 장점 : 제어기의 구현이 단순하다.
  - 단점 : CPU의 시간 낭비가 심하다.

Micro Cycle Time 부여 방식

- 동기 가변식(Synchronous Variable)
  - 수행시간이 유사한 Micro Operation끼리 그룹을 만들어, 각 그룹별로 서로 다른 Micro Cycle Time을 정의하는 방식이다.
  - 마이크로 오퍼레이션 수행시간이 현저한 차이를 나타낼 때 사용한다.
  - 동기 가변식은 동기 고정식에 비해 CPU 시간 낭비를 줄일 수 있는 반면, 제어기의 구현은 조금 복잡하다.

- 비 동기식(Asynchronous)
  - 모든 마이크로 오퍼레이션에 대하여 서로 다른 Micro Cycle Time을 정의하는 방식이다.
  - CPU의 시간 낭비는 전혀 없으나, 제어기가 매우 복잡해지기 때문에 실제로는 거의 사용되지 않는다.

기출문제

SECTION 56

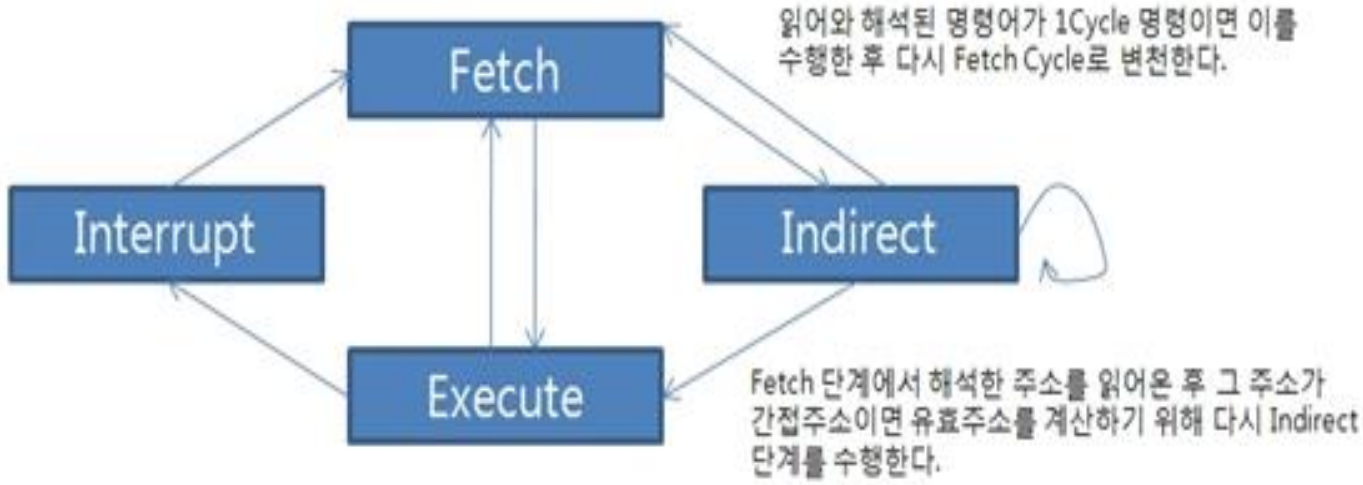
메이저 스테이트

(Major State)

메이저 스테이트의 정의

- 메이저 스테이트는 현재 CPU가 무엇을 하고 있는가를 나타내는 상태로서, CPU가 무엇을 위해 주기억장치에 접근하느냐에 따라 Fetch, Indirect, Execute, Interrupt 이렇게 4개의 상태가 있다.
- CPU는 메이저 스테이트의 네 가지 단계를 반복적으로 거치면서 동작을 수행한다.
- 메이저 스테이트는 메이저 스테이트 레지스터를 통해서 알 수 있다.
- Major Cycle 또는 Machine Cycle이라고도 한다.

플립플롭 상태		신호	메이저 스테이트
F	R		
0	0	C_0	Fetch
0	1	C_1	Indirect
1	0	C_2	Execute
1	1	C_3	Interrupt



인출 단계(Fetch Cycle)

- Fetch Cycle은 명령어를 주 기억장치에서 중앙 처리 장치의 명령 레지스터로 가져와 해독하는 단계로, 명령이 실행되기 위해서 가장 먼저 수행되는 동작이다.
- 읽어와 해석된 명령어가 1Cycle 명령이면 이를 수행한 후 다시 Fetch Cycle로 변천한다.
- 1Cycle 명령이 아니면, 해석된 명령어의 모든 비트에 따라 직접 주소와 간접 주소를 판단한다.
  - 모든 비트가 0이면 직접 주소 이므로 Execute 단계로 변천한다.
  - 모든 비트가 1이면 간접 주소 이므로 Indirect 단계로 변천한다.

제어 신호	Micro Operation	의미
C_0t_0	MAR ← PC	PC에 있는 번지를 MAR에 전송시킨다.
C_0t_1	MBR ← M[MAR], PC ← PC + 1	메모리에서 MAR이 지정하는 위치의 값을 MBR에 전송한다. 다음에 실행할 명령의 위치를 지정하기 위해 PC의 값을 1 증가 시킨다.
C_0t_2	IR ← MBR[OP], I ← MBR[I]	명령어의 OP-Code 부분을 명령 레지스터에 전송한다. 명령어의 모드 비트를 플립플롭 I에 전송한다.
C_0t_3	F ← 1 또는 R ← 1	I가 0이면 F플립플롭에 1을 전송하여 Execute 단계로 변천하고, I가 1이면 R 플립플롭에 1을 전송하여 Indirect 단계로 변천한다.

간접 단계(Indirect Cycle)

- Fetch 단계에서 해석된 명령의 주소부가 간접 주소인 경우 수행된다.
- 이 사이클에서 Fetch 단계에서 해석한 주소를 읽어온 후 그 주소가 간접주소 이면 유효 주소를 계산하기 위해 다시 Indirect 단계를 수행한다.
- 분기 같은 1Cycle 명령이면 Fetch Cycle로 변천한다.
- 실행 명령이면 Execute Cycle로 변천한다.

제어 신호	Micro Operation	의미
C_0t_0	MAR ← MBR[AD]	MBR에 있는 명령어의 번지 부분을 MAR에 전송한다.
C_0t_1	MBR ← M[MAR]	메모리에서 MAR이 지정하는 위치의 값을 MBR에 전송한다.
C_0t_2	No Operation	동작 없음
C_0t_3	F ← 1, R ← 1	F에 1, R에 0을 전송하여 Execute 단계로 변천한다.

실행 단계(Execute Cycle)

- Fetch 단계에서 인출하여 해석한 명령을 실행하는 단계이다.
- Execute 단계에서는 플래그 레지스터의 상태 변화를 검사하여 Interrupt 단계로 변천할 것인지를 판단한다.
- Execute 단계에서는 Interrupt 요청 신호를 나타내는 플래그 레지스터의 변화가 없으면 Fetch 단계로 변천한다.
- 다음은 ADD 연산을 수행하는 Execute 단계이다.

제어 신호	Micro Operation	의미
C_2t_0	MAR ← MBR[AD]	MBR에 있는 명령어의 번지 부분을 MAR에 전송한다.
C_2t_1	MBR ← M[MAR]	메모리에서 MAR이 지정하는 위치의 값을 MBR에 전송한다.
C_2t_2	AC ← AC + MBR	누산기의 값과 MBR의 값을 더해 누산기에 전송한다.
C_2t_3	F ← 1 또는 R ← 1	Execute Cycle은 F와 R의 플립플롭 상태가 F=1, R=0이다. F에 0을 전송하면 F=0, R=0이 되어 Fetch 단계로 변천하고, R에 1을 주면 F=1, R=1이 되어 Interrupt 단계로 변천한다.

인터럽트 단계 (Interrupt Cycle)

- 인터럽트 발생 시 복귀 주소를 저장시키고, 제어 순서를 인터럽트 처리 프로그램의 첫 번째 명령으로 옮기는 단계이다.
- 인터럽트 단계를 마친 후에는 항상 Fetch 단계로 변천한다.
- 다음은 Interrupt Cycle의 동작 순서이다.

제어 신호	Micro Operation	의미
C_3t_0	MBR[AD] ← PC , PC ← 0	PC가 가지고 있는, 다음에 실행할 명령의 주소를 MBR의 주소 부분으로 전송한다. 복귀 주소를 저장할 0번지를 PC에 전송한다.
C_3t_1	MBR ←PC, PC ← PC + 1	PC가 가지고 있는 값 0번지를 MAR에 전송한다. 인터럽트 처리 루틴으로 이동할 수 있는 인터럽트 벡터의 위치를 지정하기 위해 PC의 값을 1증가시켜 1로 세트시킨다.
C_3t_2	M[MAR] ← MBR , IEN ← 0	MBR이 가지고 있는 다음에 실행할 명령의 주소를 메모리의 MAR이 가르키는 위치에 저장한다. 인터럽트 단계가 끝날 때 까지 다른 인터럽트가 발생하지 않게 IEN에 0을 전송한다.
C_3t_3	F ← 0 R ← 0	F에 0, R에 0을 전송하여 Fetch 단계로 변천한다.

기출문제

SECTION 57

주요 명령의 마이크로 오퍼레이션

주요 명령의 마이크로 오퍼레이션

- AND : AC ← AC ∧ M[AD]

AND는 AC (누산기) 와 메모리의 내용을 AND (논리곱) 하여 결과를 AC에 저장하는 연산 명령이다.

제어 신호		Micro Operation	의미
C_2t_0		MAR ← MBR[AD]	MBR에 있는 명령어의 번지 부분을 MAR에 전송한다. 현재 MBR에는 Fetch 단계에서 읽어온 명령어가 들어 있다.
C_2t_1		MBR ← M[MAR]	메모리에서 MAR이 지정하는 위치의 값을 MBR에 전송한다.
C_2t_2		AC ← AC ∧ MBR	누산기의 값과 MBR의 값을 AND 연산하여 누산기에 전송한다. 실질적인 AND 연산이 이루어지는 부분이다.
C_2t_3	IEN^I	F ← 0	인터럽트 요청 신호가 없으므로, F에 0을 전송하면 F = 0, R = 0dl 되어 Fetch 단계로 변천한다.
	IEN	R ← 1	인터럽트 요청 신호가 있으므로, R에 1을 전송하면 F = 1, R = 1dl 되어 Interrupt 단계로 변천한다.

주요 명령의 마이크로 오퍼레이션

- ADD : AC ← AC + M[AD]

ADD는 AC (누산기)와 메모리의 내용을 더하여 결과를 AC에 저장하는 연산 명령이다.

제어 신호		Micro Operation	의미
C_2t_0		MAR ← MBR[AD]	MBR에 있는 명령어의 번지 부분을 MAR에 전송한다.
C_2t_1		MBR ← M[MAR]	메모리에서 MAR이 지정하는 위치의 값을 MBR에 전송한다.
C_2t_2		AC ← AC + MBR	누산기의 값과 MBR의 값을 더해 누산기에 전송한다.
C_2t_3	IEN^I	F ← 0	F에 0을 전송하면 F = 0, R = 0이 되어 Fetch 단계로 변환한다.
	IEN	R ← 1	R에 1을 전송하면 F = 1, R = 1이 되어 Interrupt 단계로 변천한다.

주요 명령의 마이크로 오퍼레이션

- LDA(Load to AC) : AC ← M[AD]

LDA는 메모리의 내용을 AC로 가져오는(Load) 명령이다.

제어 신호		Micro Operation	의미
C_2t_0		MAR ← MBR[AD]	MBR에 있는 명령어의 번지 부분을 MAR에 전송한다.
C_2t_1		MBR ← M[MAR] AC ← 0	메모리에서 MAR이 지정하는 위치의 값을 MBR에 전송한다. AC에 0을 전송하여 AC를 초기화 한다.
C_2t_2		AC ← AC + MBR	메모리에서 가져온 MBR과 AC를 더해 AC에 전송한다. 초기화된 AC에 더해지므로 메모리의 값을 AC로 불러오는 것과 같다.
C_2t_3	IEN^I	F ← 0	F에 0을 전송하면 F = 0, R = 0이 되어 Fetch 단계로 변천한다.
	IEN	R ← 1	R에 1을 전송하면 F = 1, R = 1dI 되어 Interrupt 단계로 변천한다.

주요 명령의 마이크로 오퍼레이션

- STA(Store AC) : M[AD] ← AC

STA는 AC의 내용을 메모리에 저장 하는 명령이다.

제어 신호		Micro Operation	의미
C_2t_0		MAR ← MBR [AD]	MBR에 있는 명령어의 번지 부분을 MAR에 전송한다.
C_2t_1		MBR ← AC	AC의 값을 MBR에 전송한다.
C_2t_2		M[MAR] ← MBR	MBR의 값을 메모리의 MAR이 지정하는 위치에 전송한다.
C_2t_3	IEN^I	F ← 0	F에 0을 전송하면 F = 0, R = 0이 되어 Fetch단계로 변천한다.
	IEN	R ← 1	R에 1을 전송하면 F = 1, R = 1이 되어 Interrupt 단계로 변천한다.

주요 명령의 마이크로 오퍼레이션

- BUN(Branch Unconditionally)

BUN은 PC에 특정한 주소를 전송하여 실행 명령의 위치를 변경하는 무조건 분기 명령이다.

제어 신호		Micro Operation	의미
C_2t_0		PC ← MBR[AD]	MBR에 있는 명령어의 번지 부분을 PC에 전송한다. 다음에 실행할 명령의 주소를 갖는 PC의 값이 변경되었으므로 변경된 주소에서 다음 명령이 실행된다.
C_2t_1		No Operation	동작 없음
C_2t_2		No Operation	동작 없음
C_2t_3	IEN^I	F ← 0	F에 0을 전송하면 F = 0, R = 0이 되어 Fetch 단계로 변천한다.
	IEN	R ← 1	R에 1을 전송하면 F = 1, R = 1이 되어 Interrupt 단계로 변천한다.

주요 명령의 마이크로 오퍼레이션

- BSA(Branch and Save Return Address)

BSA는 복귀 주소를 저장하고 부 프로그램을 호출하는 명령이다.

제어 신호		Micro Operation	의미
C_2t_0		MAR ← MBR[AD], MBR[AD] ← PC, PC ← MBR[AD]	MBR에 있는 명령어의 번지 부분을 MAR에 전송한다. PC의 값을 MBR의 주소 부분으로 전송한다. MBR의 주소 부분을 PC로 전소한다.
C_2t_1		M[MAR] ← MBR[AD]	MBR에 있는 명령어의 번지 부분을 메모리의 MAR이 가르키는 위치에 전송한다.
C_2t_2		PC ← PC + 1	PC의 값을 1 증가시킨다.
C_2t_3	IEN^I	F ← 0	F에 0을 전송하면 F = 0, R = 0이 되어 Fetch 단계로 변천한다.
	IEN	R ← 1	R에 1을 전송하면 F = 1, R = 1이 되어 Interrupt 단계로 변천한다.

주요 명령의 마이크로 오퍼레이션

- ISZ(Increment and Skip if Zero)

ISZ는 메모리의 값을 읽어 그 값을 1 증가시킨 후 음수에서 시작한 그 값이 0이면 현재 명령을 건너 띄어 다음 명령으로 이동한다.

제어 신호	Micro Operation	의미
C_2t_0	MAR ← MBR[AD]	MBR에 있는 명령어의 번지 부분을 MAR에 전송한다.
C_2t_1	MBR ← M[MAR]	메모리에서 MAR이 지정하는 위치의 값을 MBR에 전송한다.
C_2t_2	MBR ← MBR + 1	MBR의 값을 1 증가시킨다.
C_2t_3	M[MAR] ← MBR, IF(MBR = 0) THEN ← PC + 1	MBR의 값을 메모리의 MAR이 지정하는 위치에 전송한다. MBR의 값이 0이면 다음 명령을 수행한다.

SECTION 58

제어기의 구현

제어 데이터

- 제어 데이터는 제어 장치가 제어 신호를 발생하기 위한 자료로서, CPU가 특정한 메이저 상태와 타이밍 상태에 있을 때 제어 자료에 따른 제어 규칙에 의해 제어 신호가 발생한다.

- 제어 데이터 종류

- 메이저 스테이트 사이의 변천을 제어하는 데이터
- 중앙처리장치의 제어점을 제어하는 데이터
- 인스트럭션의 수행 순서를 결정하는 데 필요한 제어 데이터

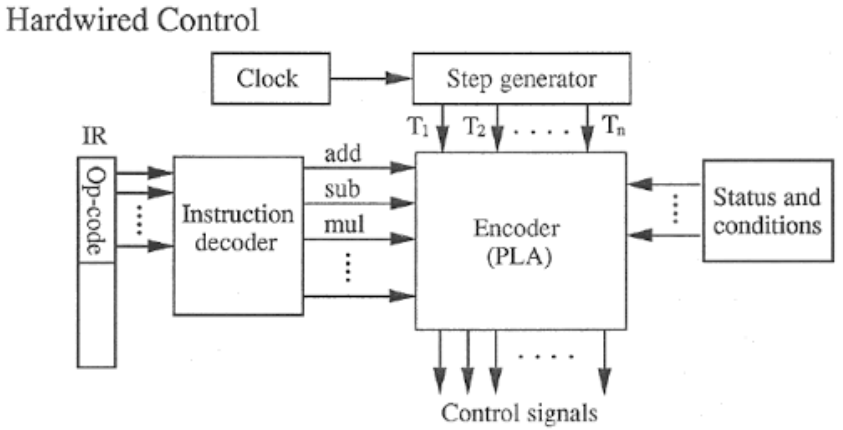
구분	Fetch	Indirect	Execute	Interrupt
State 간 변이용	명령어 종류 주소지정방식	주소 지정 방식	인터럽트 요청 신청	없음
제어점 제어용	명령어	유효 주소	명령어의 연산자	Interrupt 체제에 따라 달라짐
수행 순서 제어용	PC	없음	PC	Interrupt 체제에 따라 달라짐

제어기의 구현

- 제어기 (제어 장치)는 필요한 마이크로 연산들이 연속적으로 수행할 수 있도록 제어 신호를 보내는 역할을 하는 장치를 말한다. 제어 장치는 하드웨어적으로 구현하는 고정 배선 제어 장치와 소프트웨어적으로 구현하는 마이크로 프로그래밍 기법이 있다.

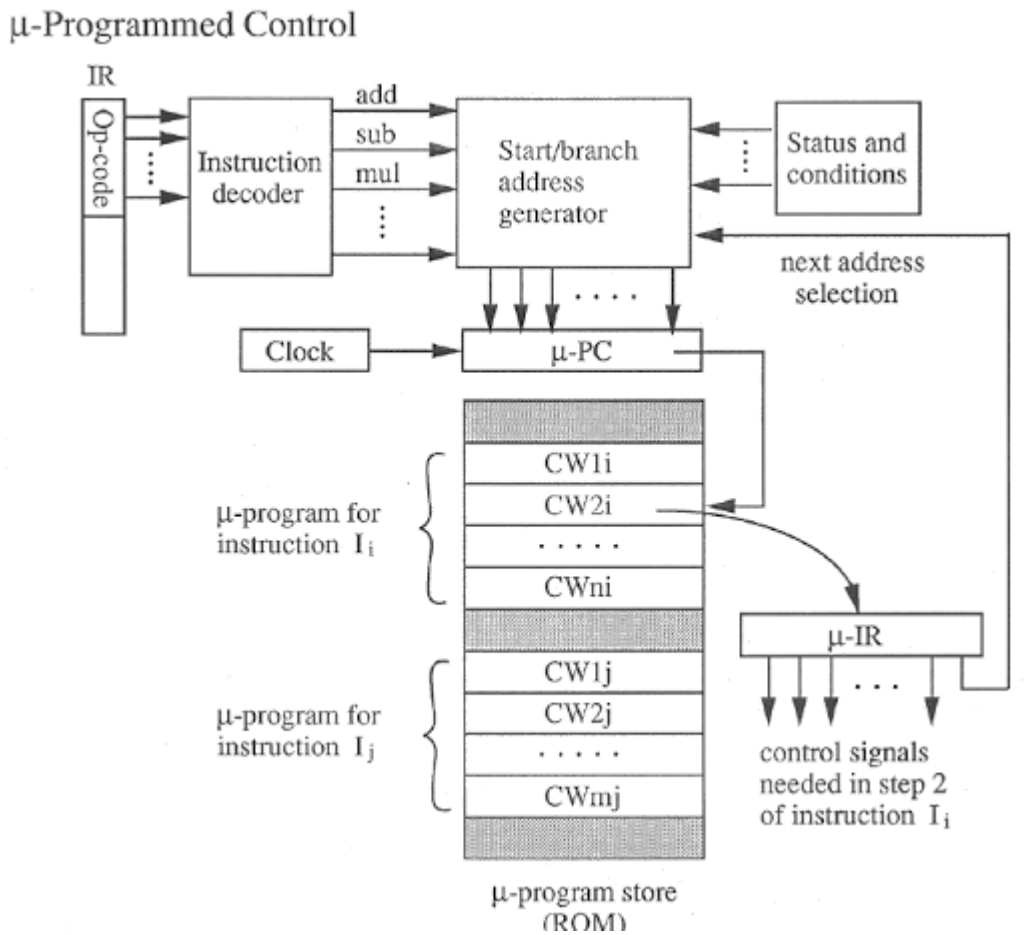
- 고정 배선 제어장치(Hard-wired Control Unit)

- 각 독립 제어점에 제어신호를 가해야 할 조건들을 제어 데이터와 제어기의 상태로 표현한 후 이를 만족하는 조합논리회로를 설계하여 해당 제어점에 연결하는 방식.
- Hardware적인 구성 방법이다.
- 속도가 빠르다.
- 마이크로 프로그램 방식에 비해 비싸다.
- RISC 구조를 기본으로 하는 컴퓨터에서 주로 사용된다.
- 한 번 만들어진 명령어 세트를 변경할 수 없다.
- 회로 구성이 복잡하다.



마이크로 프로그래밍 기법(Micro Programmed Control Unit)

- 마이크로 프로그램은 내부 제어 신호를 지정하는 여러 가지 마이크로 인스트럭션으로 작성하는 것.
- Software적인 구성 방법이다. 정확히 말하면 펌웨어를 이용하는 방식이다.
- 마이크로 프로그램된 제어장치를 사용하는 컴퓨터는 주 메모리 외에 마이크로 프로그램이 저장되는 제어 메모리가 필요하다.
- 마이크로 프로그램을 이용하여 명령어 세트를 쉽게 변경할 수 있다.
- 다양한 어드레스 모드를 갖는다.
- 하드웨어 방식에 비해 속도가 느리다.
- 유지보수 및 수정이 용이하다.
- 비교적 복잡한 명령 세트를 가진 시스템에 적합하다.



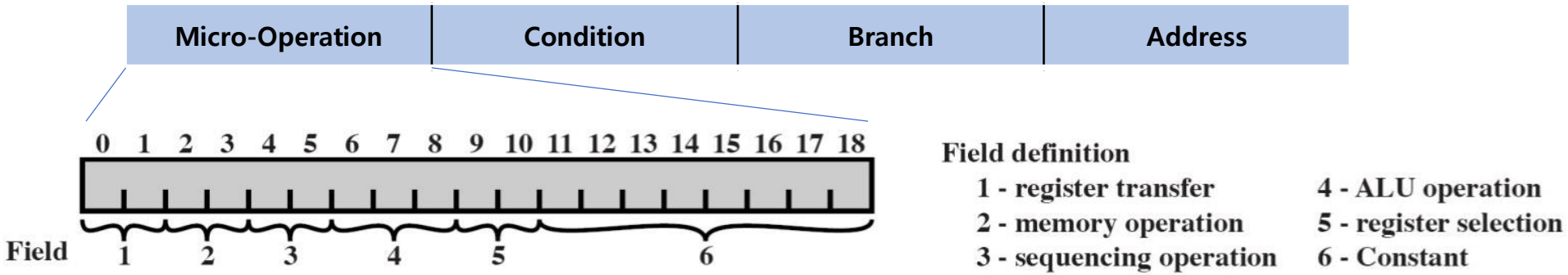
제어장치의 특징 비교

구분	고정배선 제어장치	마이크로 프로그래밍 기법
반응 속도	고속	저속
회로 복잡도	복잡	간단
경제성	비경제적	경제적
융통성	없음	있음
구성	하드웨어	소프트웨어

- 마이크로 프로그램에서 제어 메모리의 번지 결정
- 제어 메모리에서는 계속적인 마이크로 명령어를 수행하기 위하여 다음과 같은 방향으로 다음에 실행할 제어 메모리의 번지를 결정한다.
- 제어 주소 레지스터(CAR) : 값을 1 증가
- 명령 레지스터(IR) : 지정하는 번지로 무조건 분기, 주소 필드로부터 제어 메모리의 주소로 매핑(Mapping)
- 상대 레지스터(SR) : 조건에 따른 조건부 분기
- 서브루틴의 Call과 Return

마이크로 명령의 형식

- **수평 마이크로 명령(Horizontal Micro Instruction)**
  - 마이크로 명령의 한 비트가 한 개의 마이크로 동작을 관할하는 명령이다.
  - Micro Operation부가 m Bit일 때 m개의 마이크로 동작을 표현할 수 있다.

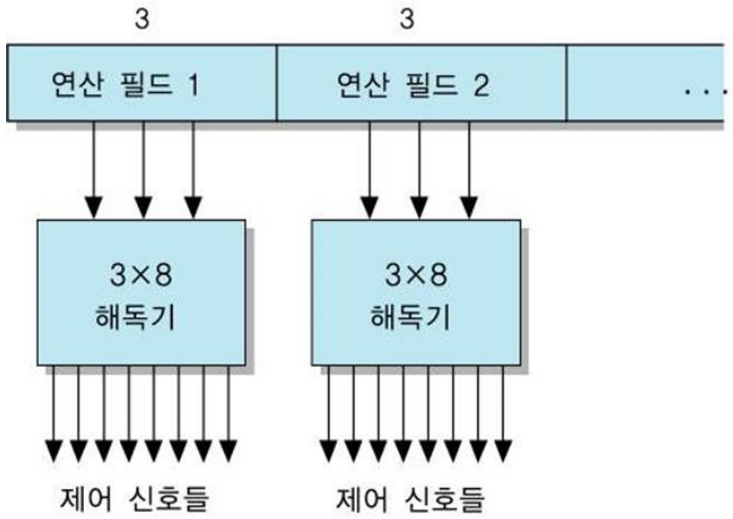


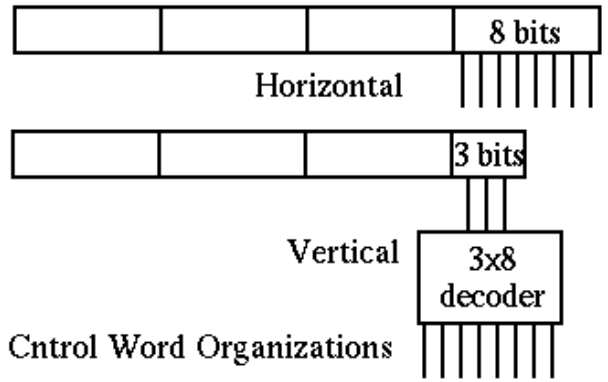
- 장점 : 각 비트가 하나의 마이크로 동작을 제어하기 때문에 제어 비트를 디코딩할 필요가 없고, 마이크로 명령 한 개로 여러 개의 하드웨어 구성 요소를 동시에 동작시킬 수 있다.
- 단점 : 제어 워드의 비트들이 충분히 활용되지 못하며, 제어 워드의 길이가 길어지기 때문에 비용이 많이 든다.

마이크로 명령의 형식

수직 마이크로 명령(Vertical Micro Instruction)

- 제어 메모리 외부에서 디코딩 회로를 필요로 하는 마이크로 명령이다.
- 디코더의 출력을 제어 신호로 사용한다.
- 한 개의 마이크로 명령으로 한 개의 마이크로 동작만 제어할 수 있다.





마이크로 명령의 형식

- **나노 명령(Nano Instruction)**

- 나노 메모리라는 낮은 레벨의 메모리에 저장된 마이크로 명령을 나노 명령이라 한다.
- 나노 명령은 수직 마이크로 명령을 수행하는 제어기에서 디코더를 ROM (나노 메모리)로 대치하여 두 메모리 레벨로 구성한다.
- 제어 메모리의 각 Word에는 나노 명령이 저장되어 있는 나노 메모리의 번지들을 저장하고 있다.
- 제어 메모리에서 인출한 번지에 해당하는 나노 메모리 내의 기억 장소를 지정하여 그곳의 나노 명령을 인출해서 마이크로 동작을 제어한다.

기출문제