

CPM의 구성요소: ALU, 제어유닛, 레지스터

# 컴퓨터구조론

Computer Architecture

## 제4장 제어 유닛

박숙영

blue@sookmyung.ac.kr

- 
- 4.1 제어 유닛의 기능
  - 4.2 제어 유닛의 구조
  - 4.3 마이크로 명령어의 형식
  - 4.4 마이크로프로그래밍
  - 4.5 마이크로프로그램의 순서 제어

## 4.1 제어 유닛의 기능

---

### ■ 제어 유닛의 기능

- 명령어 코드의 해독
  - 명령어 실행에 필요한 제어 신호들의 발생
- 마이크로명령어(micro-instruction) : 명령어 사이클의 각 주기에서 실행되는 각 마이크로-연산을 지정해주는 2진 비트들로서, 제어 단어(control word)라고도 함
- 마이크로프로그램(microprogram) : 마이크로명령어들의 집합
- 루틴(routine) : CPU의 특정 기능을 수행하기 위한 마이크로명령어들의 그룹
- [예] 인출 사이클 루틴, 실행 사이클 루틴, 인터럽트 사이클 루틴

서보시스템의 개념

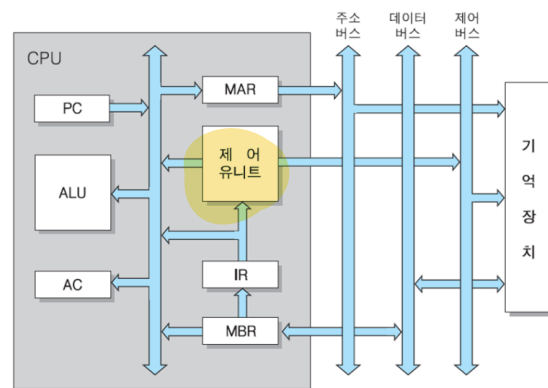
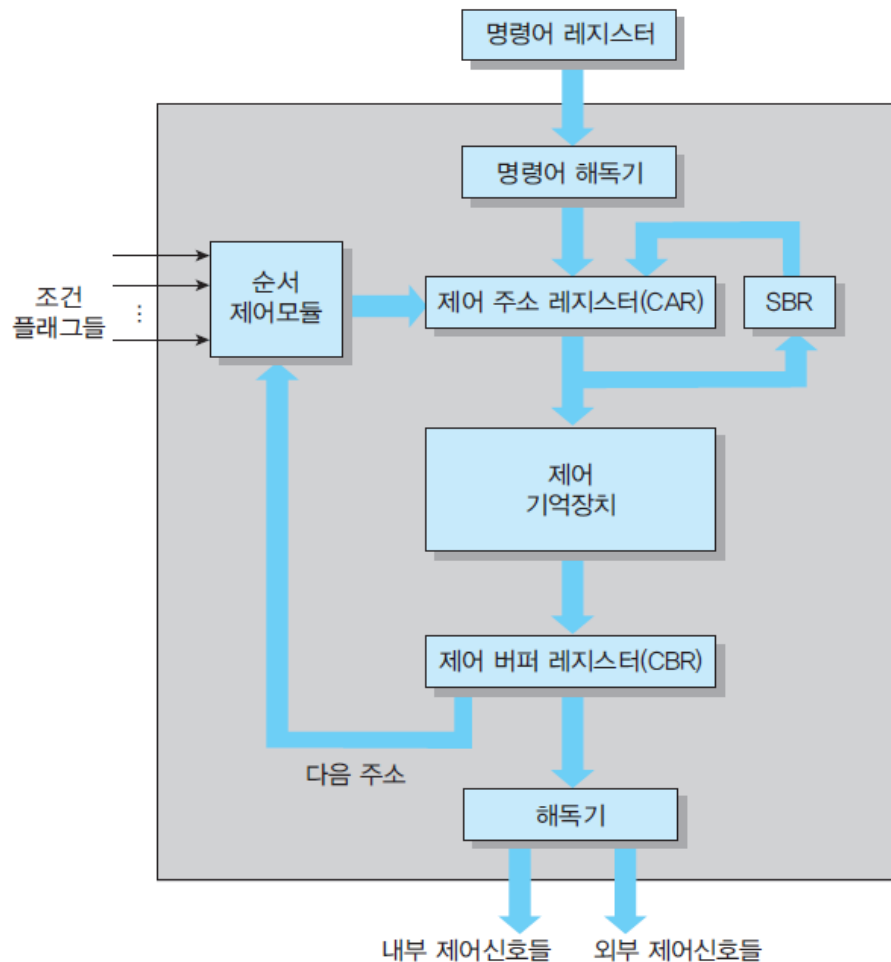
## 4.2 제어 유닛의 구조

### ■ 구성 요소들

- 명령어 해독기(instruction decoder) : 명령어 레지스터(IR)로부터 들어오는 명령어의 연산 코드를 해독하여 해당 연산을 수행하기 위한 루틴의 시작 주소를 결정
- 제어 주소 레지스터(control address register: CAR) : 다음에 실행할 마이크로 명령어의 주소를 저장하는 레지스터  
→ 이 주소는 제어 기억장치의 특정 위치를 지칭  
↳ ROM에 해당함
- 제어 기억장치(control memory) : 마이크로명령어들로 이루어진 마이크로프로그램을 저장하는 내부 기억장치

read-only ROM 형태

# 제어 유닛의 내부 구성도



[참고: CPU 내부 구조]

## 제어 유닛의 구조 (계속)

---

- 제어 버퍼 레지스터(control buffer register: CBR) : 제어 기억장치로부터 읽혀진 마이크로명령어 비트들을 일시적으로 저장하는 레지스터
- 서브루틴 레지스터(subroutine register: SBR) : 마이크로프로그램에서 서브루틴이 호출되는 경우에 현재의 CAR 내용을 일시적으로 저장하는 레지스터
- 순서제어 모듈(sequencing module) : 마이크로명령어의 실행 순서를 결정하는 회로들의 집합

# CPU의 명령어 세트 설계 및 구현 과정

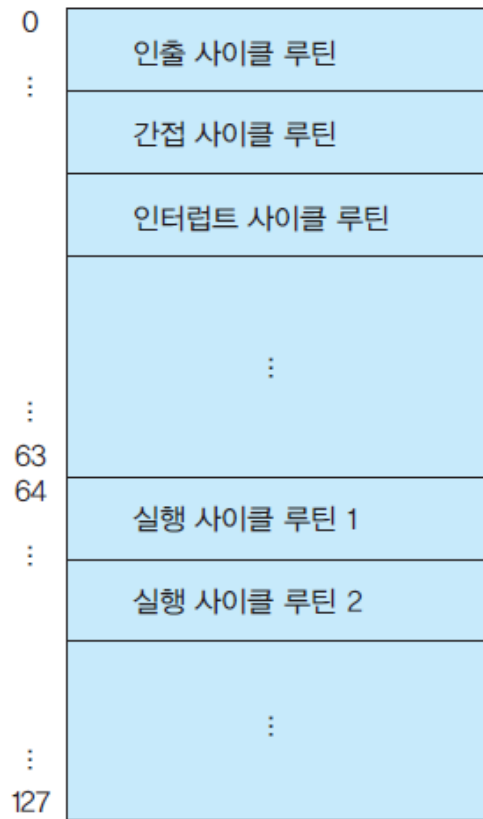
---

- 명령어들의 종류 및 비트 패턴 정의
- 명령어들의 실행에 필요한 하드웨어 설계
- 각 명령어를 위한 실행 사이클 루틴 작성(마이크로프로그래밍)
- 마이크로프로그램 코드들을 제어 기억장치에 저장

# 제어 기억장치의 내부 구성

## ■ 마이크로프로그램 루틴들을 제어 기억장치에 저장한 예

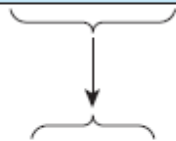
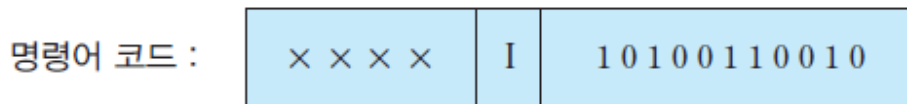
- 제어 기억장치 용량 = 128 단어
- 전반부 (0 ~ 63번지) : 공통 루틴들 저장
- 후반부 (64 ~ 127번지) : 각 명령어의 실행 사이클 루틴들 저장





# 명령어 해독 과정

- 명령어 해독: 명령어의 연산 코드가 지정하는 연산을 위한 실행 사이클 루틴의 시작 주소를 결정하는 동작
- 사상(mapping)을 이용한 해독 방법
  - 명령어의 연산 코드를 특정 비트 패턴과 조합
  - [예] 16-비트 길이의 명령어가 4 비트의 연산 코드, 1 비트의 간접 주소지정(I) 비트 및 11 비트의 주소로 구성된 경우



- 연산 코드      사상 함수 :      1 × × × × 0 0      (6810)
- 연산 코드 = 0110 → 실행 사이클 루틴의 시작 주소 = 1011000 (8810)

## 4.3 마이크로명령어의 형식

- 연산 필드가 두 개이면, 두 개의 마이크로-연산들을 동시에 수행 가능
- 조건(CD) 필드는 분기에 사용될 조건 플래그를 지정
- 분기(BR) 필드는 분기의 종류와 다음에 실행할 마이크로명령어의 주소를 결정하는 방법을 명시
- 주소 필드(ADF)의 내용은 분기가 발생하는 경우에 목적지 마이크로명령어의 주소로 사용

|         |         |       |       |            |
|---------|---------|-------|-------|------------|
| 3       | 3       | 2     | 2     | 7          |
| 연산 필드 1 | 연산 필드 2 | 조건 필드 | 분기 필드 | 주소 필드(ADF) |

# 마이크로연산들에 대한 2진 코드 및 기호 [예]

- '연산필드 1'에 위치할 마이크로-연산들

| 코드  | 마이크로-연산                   | 기호    |
|-----|---------------------------|-------|
| 000 | None                      | NOP   |
| 001 | $MAR \leftarrow PC$       | PCTAR |
| 010 | $MAR \leftarrow IR(addr)$ | IRTAR |
| 011 | $AC \leftarrow AC + MBR$  | ADD   |
| 100 | $MBR \leftarrow M[MAR]$   | READ  |
| 101 | $AC \leftarrow MBR$       | BRTAC |
| 110 | $IR \leftarrow MBR$       | BRTIR |
| 111 | $M[MAR] \leftarrow MBR$   | WRITE |

## 마이크로 연산들에 대한 2진 코드 및 기호 [예] (계속)

- '연산필드 2'에 위치할 마이크로-연산들

| 코드  | 마이크로-연산                  | 기호    |
|-----|--------------------------|-------|
| 000 | None                     | NOP   |
| 001 | $PC \leftarrow PC + 1$   | INCPC |
| 010 | $MBR \leftarrow AC$      | ACTBR |
| 011 | $MBR \leftarrow PC$      | PCTBR |
| 100 | $PC \leftarrow MBR$      | BRTPC |
| 101 | $MAR \leftarrow SP$      | SPTAR |
| 110 | $AC \leftarrow AC - MBR$ | SUB   |
| 111 | $PC \leftarrow IR(addr)$ | IRTPC |

# 조건 필드의 코드 지정

- 조건 필드 : 두 비트로 구성되며, 분기의 조건으로 사용
  - U : 무조건 분기
  - I : 만약 I = 1이면, 간접 사이클 루틴을 호출
  - S : 누산기에 저장된 데이터의 부호가 1이면, 분기
  - Z : 누산기에 저장된 데이터가 0 (Z=1)이라면, 분기

| 코드 | 조건    | 기호 | 설명                   |
|----|-------|----|----------------------|
| 00 | 1     | U  | 무조건 분기               |
| 01 | I 비트  | I  | 간접 주소지정              |
| 10 | AC(S) | S  | 누산기(AC)에 저장된 데이터의 부호 |
| 11 | AC=0  | Z  | AC에 저장된 데이터 = 0      |

# 분기 필드의 코드 지정

## ■ 분기 필드 : 두 비트로 구성되며, 분기의 유형을 지정

- 조건 필드의 조건이 만족되면, ADF 필드의 내용을 CAR로 적재  
→ 그 주소로 분기 (JUMP 혹은 CALL)
- RET : 서브루틴으로부터 복귀(SBR에 저장된 내용을 CAR로 적재)
- MAP : 사상 방식에 의하여 분기 목적지 주소 결정

| 코드 | 기호   | 설명  |
|----|------|---|
| 00 | JMP  | 만약 조건 = 1이면, $CAR \leftarrow ADF$<br>만약 조건 = 0이면, $CAR \leftarrow CAR + 1$                            |
| 01 | CALL | 만약 조건 = 1이면, $SBR \leftarrow CAR + 1$ , $CAR \leftarrow ADF$<br>만약 조건 = 0이면, $CAR \leftarrow CAR + 1$ |
| 10 | RET  | $CAR \leftarrow SBR$ (서브루틴으로부터의 복귀)   |
| 11 | MAP  | $CAR(1) \leftarrow 1$ , $CAR(2-5) \leftarrow IR(op)$ , $CAR(6,7) \leftarrow 0$                        |

## 4.4 마이크로프로그래밍

### 4.4.1 인출 사이클 루틴의 마이크로명령어 루틴

ORG 0

FETCH : PCTAR            U JMP NEXT ; MAR ← PC, 다음 마이크로명령어 실행  
          READ, INCPC    U JMP NEXT ; MBR ← M[MAR], PC = PC + 1,  
  ; 다음 마이크로명령어 실행

          BRTIR            U MAP            ; IR ← MBR,  
  ; 해당 실행 사이클 루틴으로 분기

- 2진 비트 패턴

| 주소      | $\mu$ -ops | CD | BR | ADF     |
|---------|------------|----|----|---------|
| 0000000 | 001 000    | 00 | 00 | 0000001 |
| 0000001 | 100 001    | 00 | 00 | 0000010 |
| 0000010 | 110 000    | 00 | 11 | 0000000 |

- 주소: 각 마이크로명령어가 저장될 제어 기억장치 주소,  $\mu$ -ops: 두 개의 마이크로-연산들, CD: 조건 필드, BR: 분기 필드, ADF: 주소 필드

## 4.4.2 간접 사이클 루틴

```
ORG 4
INDRT :  IRTAR      U JMP NEXT    ; MAR ← IR(addr),
                                           ; 다음 마이크로명령어 실행
READ      U JMP NEXT    ; MBR ← M[MAR],
                                           ; 다음 마이크로명령어 실행
BRTIR      U RET        ; IR(addr) ← MBR,
                                           ; 실행 사이클 루틴으로 복귀
```

### ■ 2진 비트 패턴

| 주소      | $\mu$ -ops | CD | BR | ADF     |
|---------|------------|----|----|---------|
| 0000100 | 010 000    | 00 | 00 | 0000101 |
| 0000101 | 100 000    | 00 | 00 | 0000110 |
| 0000110 | 110 000    | 00 | 10 | 0000000 |



### 4.4.3 실행 사이클 루틴

- 사상 방식을 이용하여 각 연산 코드에 대한 실행 사이클 루틴의 시작 주소를 결정하고, 각 명령어 실행을 위한 루틴을 작성
- 각 연산 코드에 대한 사상의 결과

| 명령어      | 연산 코드 | 루틴의 시작 주소           |
|----------|-------|---------------------|
| NOP      | 0000  | $1000000 = 64_{10}$ |
| LOAD(I)  | 0001  | $1000100 = 68_{10}$ |
| STORE(I) | 0010  | $1001000 = 72_{10}$ |
| ADD      | 0011  | $1001100 = 76_{10}$ |
| SUB      | 0100  | $1010000 = 80_{10}$ |
| JUMP     | 0101  | $1010100 = 84_{10}$ |

# 각 명령어에 대한 실행 사이클 루틴들

연산자  
그냥 넘어감

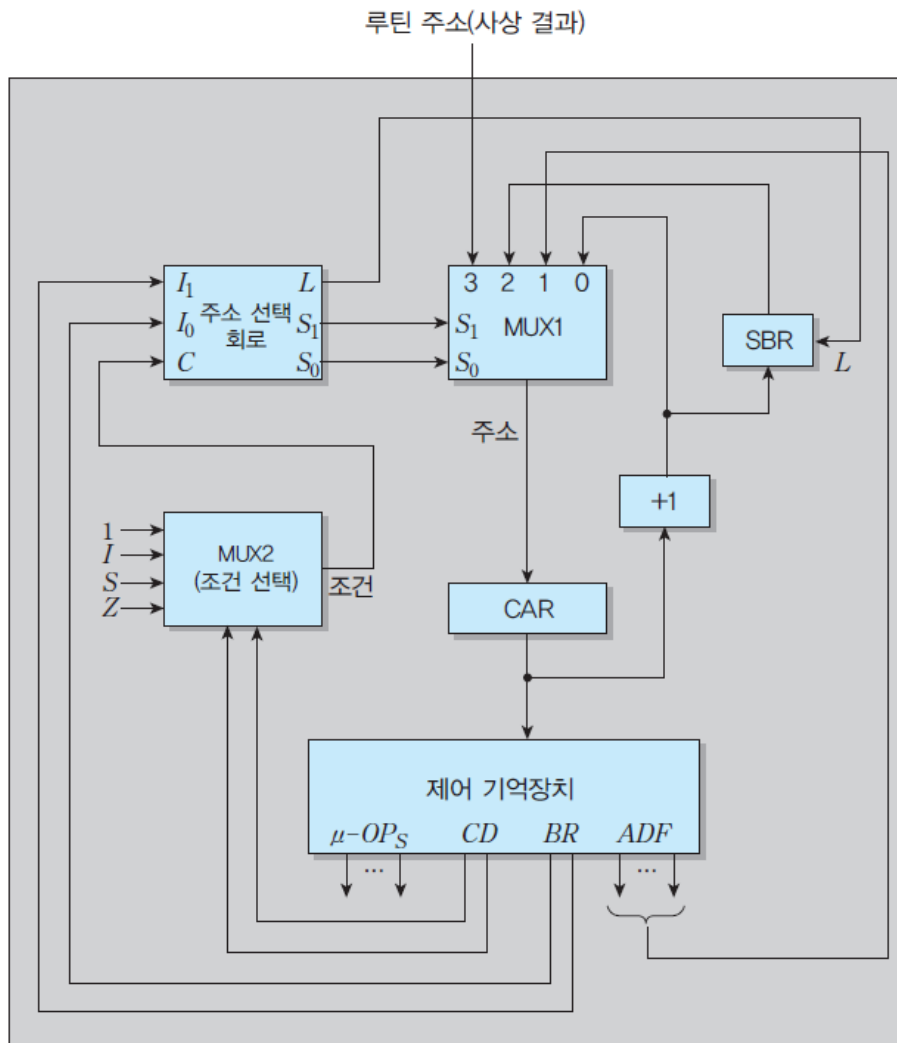
|               |        |   |      |       |                       |
|---------------|--------|---|------|-------|-----------------------|
| <u>NOP :</u>  | ORG 64 |   |      |       |                       |
|               | INCP   | U | JMP  | FETCH | ; PC ← PC+1           |
| <u>LOAD :</u> | ORG 68 |   |      |       |                       |
|               | NOP    | I | CALL | INDRT | ; I=1이면, 간접 사이클 루틴 호출 |
|               | IRTAR  | U | JMP  | NEXT  | ; MAR ← IR(addr)      |
|               | READ   | U | JMP  | NEXT  | ; MBR ← M[MAR]        |
|               | BRTAC  | U | JMP  | FETCH | ; AC ← MBR            |
| STORE :       | ORG 72 |   |      |       |                       |
|               | NOP    | I | CALL | INDRT | ; I=1이면, 간접 사이클 루틴 호출 |
|               | IRTAR  | U | JMP  | NEXT  | ; MAR ← IR(addr)      |
|               | ACTBR  | U | JMP  | NEXT  | ; MBR ← AC            |
|               | WRITE  | U | JMP  | FETCH | ; M[MAR] ← MBR        |
| ADD :         | ORG 76 |   |      |       |                       |
|               | IRTAR  | U | JMP  | NEXT  | ; MAR ← IR(addr)      |
|               | READ   | U | JMP  | NEXT  | ; MBR ← M[MAR]        |
|               | ADD    | U | JMP  | FETCH | ; AC ← AC + MBR       |
| SUB :         | ORG 80 |   |      |       |                       |
|               | IRTAR  | U | JMP  | NEXT  | ; MAR ← IR(addr)      |
|               | READ   | U | JMP  | NEXT  | ; MBR ← M[MAR]        |
|               | SUB    | U | JMP  | FETCH | ; AC ← AC - MBR       |
| JUMP :        | ORG 84 |   |      |       |                       |
|               | IRTPC  | U | JMP  | FETCH | ; PC ← IR(addr)       |

## 4.5 마이크로프로그램의 순서제어

---

- 순서제어(sequencing) : 다음에 실행할 마이크로명령어의 주소 결정
- CAR의 초기값 = 0 (인출 사이클 루틴의 첫 번째 마이크로명령어의 주소)
- MUX1 : 다음에 실행할 마이크로명령어의 주소 선택
- MUX2 : 조건 플래그를 선택하여 주소선택 회로로 전송

# 순서제어 회로가 포함된 제어 유닛의 구성도



# 주소 선택 방법

---

## ■ 주소 선택 방법

- BR = 00 (JUMP) 혹은 01 (CALL)일 때,
  - C = 0, 다음 위치의 마이크로명령어 선택
  - C = 1, 주소 필드(ADF)가 지정하는 위치로 점프(jump) 혹은 호출(call)  
(단, 호출 시에는 CAR 내용을 SBR에 저장)
- BR = 10 (RET)일 때는 SBR 내용을 CAR로 적재 : 복귀
- BR = 11 (MAP)일 때는 사상 결과를 CAR에 적재

# 주소 선택 회로의 입력 및 출력 신호들

| BR 조건 |       |     | MUX1 선택 |       | SBR | CAR로 적재될<br>MUX1의 입력 | 설명  |
|-------|-------|-----|---------|-------|-----|----------------------|---|
| $h$   | $l_0$ | $C$ | $S_1$   | $S_0$ | $L$ |                      |   |
| 0     | 0     | 0   | 0       | 0     | 0   | 0                    | $CAR \leftarrow CAR + 1$                                |
| 0     | 0     | 1   | 0       | 1     | 0   | 1                    | $CAR \leftarrow ADF \langle \text{Jump} \rangle$        |
| 0     | 1     | 0   | 0       | 0     | 0   | 0                    | $CAR \leftarrow CAR + 1$                                |
| 0     | 1     | 1   | 0       | 1     | 1   | 1                    | $SBR \leftarrow CAR + 1, CAR \leftarrow ADF$            |
| 1     | 0     | x   | 1       | 0     | 0   | 2                    | $CAR \leftarrow SBR \langle \text{Return} \rangle$      |
| 1     | 1     | x   | 1       | 1     | 0   | 3                    | $CAR \leftarrow 1XXXX00 \langle \text{Mapping} \rangle$ |

# 제어 신호의 생성

---

- 제어 기억장치로부터 인출된 마이크로명령어 내 연산 필드의 비트들이 제어 유닛의 외부로 출력되어, 각각 제어 신호로 사용됨

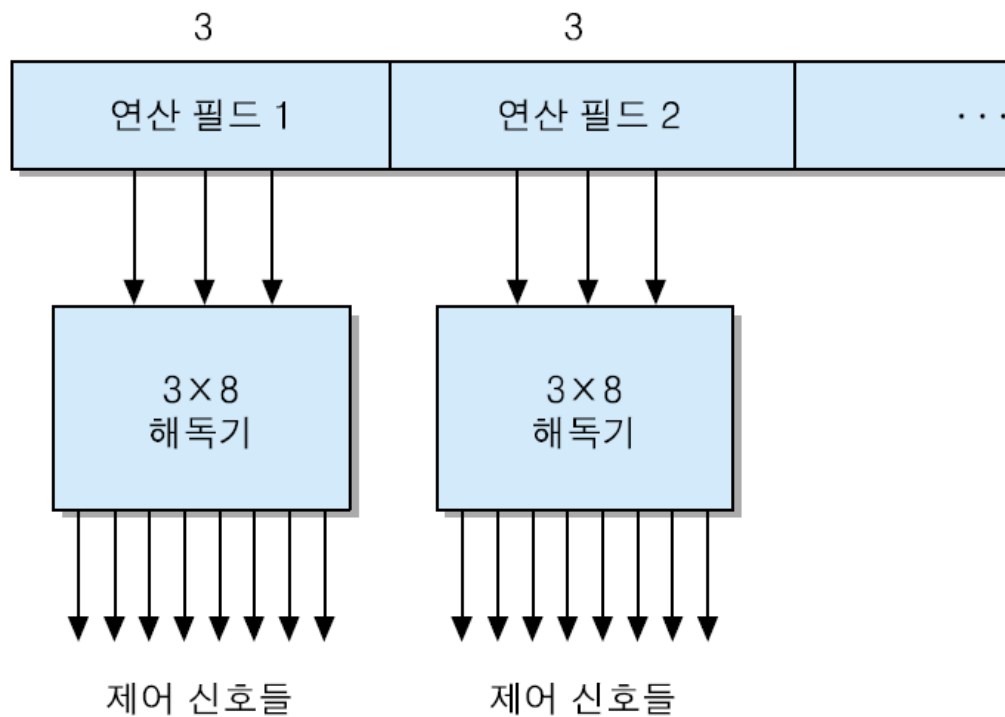
## (1) 수직적 마이크로프로그래밍

---

- Vertical microprogramming
- 마이크로명령어의 연산 필드에 적은 수의 코드화된 비트들(encoded bits)을 포함시키고, 해독기를 이용하여 그 비트들을 필요한 수 만큼의 제어 신호들로 확장하는 방식
- [장점] 마이크로명령어의 길이(비트 수) 최소화  
→ 제어 기억장치 용량 감소
- [단점] 해독 동작에 걸리는 만큼의 지연 시간 발생



# 수직적 마이크로프로그래밍에서의 제어 신호 발생 방법



## (2) 수평적 마이크로프로그래밍

---

- Horizontal microprogramming
- 연산 필드의 각 비트와 제어 신호를 1:1로 대응시키는 방식
- 필요한 제어 신호 수만큼의 비트들로 이루어진 연산필드 비트들이 마이크로명령어에 포함되어야 함
- [장점] 하드웨어가 간단하고, 해독에 따른 지연 시간이 없음
- [단점] 마이크로명령어의 비트 수가 길어지기 때문에 제어 기억장치의 용량이 증가