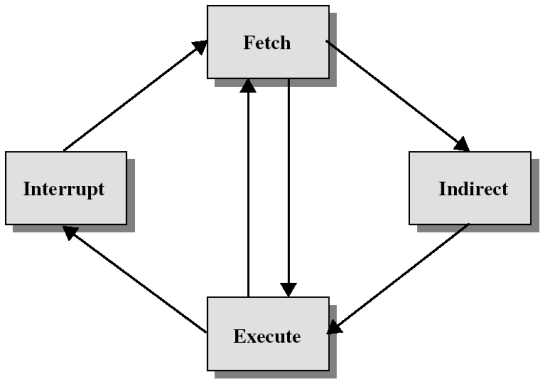
4장. 제어 유니트

목차

4.1 제어 유니트의 기능

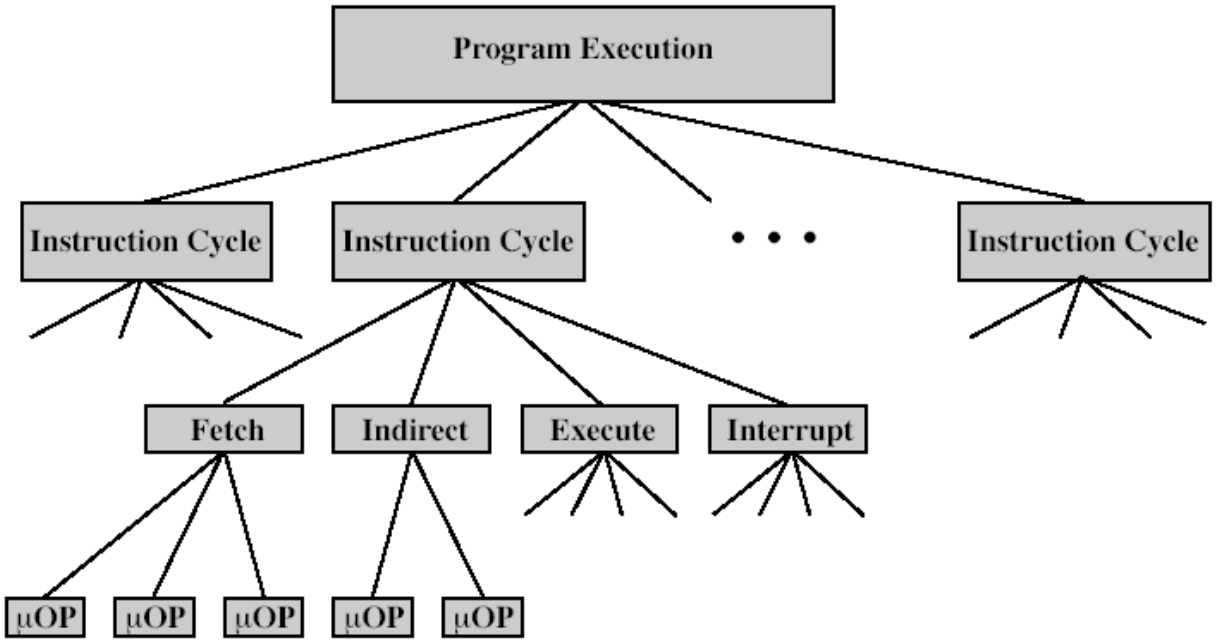
- 제어 유니트의 기능
  - 명령어 코드의 해독
  - 명령어 실행에 필요한 제어 신호들의 발생
  - 명령어 사이클이 수행되도록 제어
    - 명령어 사이클: 인출 사이클, 간접 사이클, 실행 사이클, 인터럽트 사이클



- 각 사이클은 여러 개의 마이크로-연산(micro-operation)으로 수행
  - 프로세서 레지스터를 포함하는 일련의 단계로 구성 (CPU의 원자적/기능적 연산)
  - 예) 인출 사이클의 마이크로-연산
    - » t0: MAR←PC
    - » t1: MBR←M[MAR], PC←PC+1
    - » t2: IR←MBR

프로그램 실행의 구성

- 프로그램의 실행
  - 명령어 사이클의 반복
    - 서브 사이클 (인출, 간접, 실행, 인터럽트)
      - 마이크로 연산



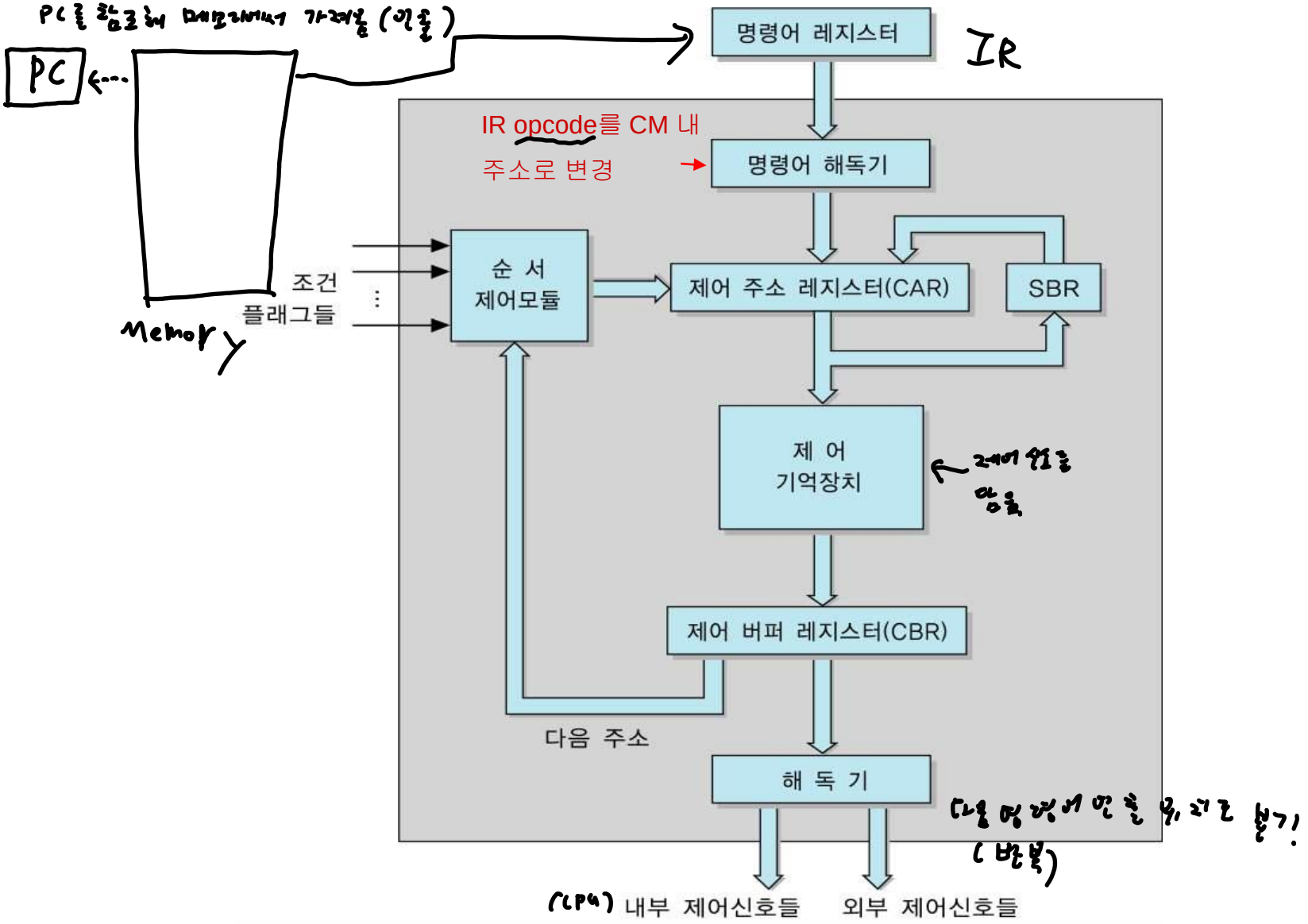
마이크로 명령어, 루틴, 마이크로 프로그램

- 마이크로 명령어(micro-instruction)
  - 명령어 사이클의 각 주기에서 실행되는 마이크로-연산들에 대응되는 비트들로 이루어진 단어
  - =제어 단어(control word)
- 마이크로 프로그램(micro-program)
  - 마이크로 명령어들의 집합
  - firmware 펌웨어, 하드-소프트 사이에 중간 역할. 실제는 프로그램 but 하드웨어 레벨
- 루틴(routine)
  - CPU의 특정 기능을 수행하기 위한 마이크로 명령어들의 집합
  - [예]
    - 인출 사이클 루틴: 모든 명령어에 공통
    - 실행 사이클 루틴
    - 인터럽트 사이클 루틴

4.2 제어 유니트의 구조

- 제어 유니트의 구성 요소
  - 명령어 해독기(instruction decoder)
    - 명령어 레지스터(IR)로부터 들어오는 명령어의 연산 코드를 해독
    - 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
  - 제어 기억장치(control memory)
    - 마이크로 명령어들로 이루어진 마이크로 프로그램을 저장하는 내부 기억장치
  - 제어 버퍼 레지스터(control buffer register: CBR)
    - 제어 기억장치로부터 읽혀진 마이크로 명령어 비트들을 일시적으로 저장하는 레지스터 (마이크로 명령어 읽기= 마이크로 명령어 실행)
  - 서브루틴 레지스터(subroutine register: SBR)
    - 마이크로 프로그램에서 서브루틴이 호출되는 경우에 현재의 CAR 내용을 일시적으로 저장하는 레지스터
  - 순서제어 모듈(sequencing module)
    - 마이크로 명령어의 실행 순서를 결정하는 회로들의 집합

제어 유니트의 내부 구성도



CPU의 명령어 세트 설계 과정

- 명령어들의 종류와 비트 패턴 정의
- 명령어들의 실행에 필요한 하드웨어 설계
- 각 명령어를 위한 실행 사이클 루틴 작성(마이크로 프로그래밍)
- 마이크로프로그램 코드들을 제어 기억장치에 저장

- 마이크로 프로그램
  - CPU 설계 단계에서 확정, 불변
  - 코드들을 제어 기억장치(ROM)에 저장하여 CPU에 삽입

제어 기억장치의 내부 구성

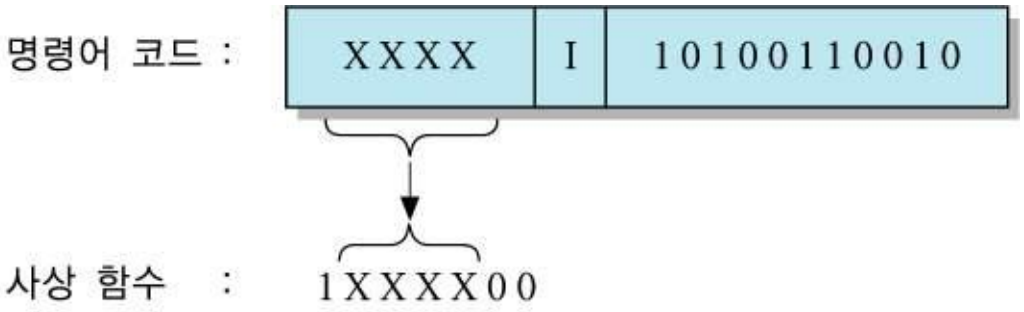
- 마이크로 프로그램 루틴들이 제어 기억장치에 저장 (4.4절)
  - 각 루틴 내의 마이크로 명령어들은 순차적으로 실행
  - 각 루틴은 다음 수행 위치를 나타내는 분기 또는 점프 명령어로 끝남
- 마이크로 프로그램 루틴들을 제어 기억장치에 저장한 예
  - 제어 기억장치 용량
    - 128 단어인 경우
  - 전반부 (0 ~ 63번지)
    - 공통 루틴들 저장
      - 인출사이클 루틴
      - 간접사이클 루틴
      - 인터럽트 사이클 루틴
  - 후반부 (64 ~ 127번지)
    - 각 명령어의 실행 사이클 루틴들 저장

명령어 해독

- 명령어의 연산 코드가 지정하는 연산을 위한 실행 사이클 루틴의 시작 주소를 결정하는 동작
- 명령어 해독의 과정
  - 인출사이클
    - 명령어 레지스터(IR)에 적재된 명령어 비트 중 연산코드가 제어유니트의 명령어 해독기로 입력 (현재까지 컴퓨터는 명령어의 동작 방법을 모름)
  - 명령어 해독(@명령어 해독기)
    - 명령어의 연산 코드가 지정하는 연산을 위한 실행 사이클 루틴의 시작 주소를 결정하는 동작 (명령어의 동작과정이 명시된 루틴을 호출하기 위해)
    - 해독 방법: 사상(mapping)을 이용하여 루틴의 시작 주소 결정
  - 실행사이클
    - 위 연산코드가 지정하는 연산이 제어 기억장치에 저장된 해당 루틴을 실행함으로써 수행

사상을 이용한 명령어 해독

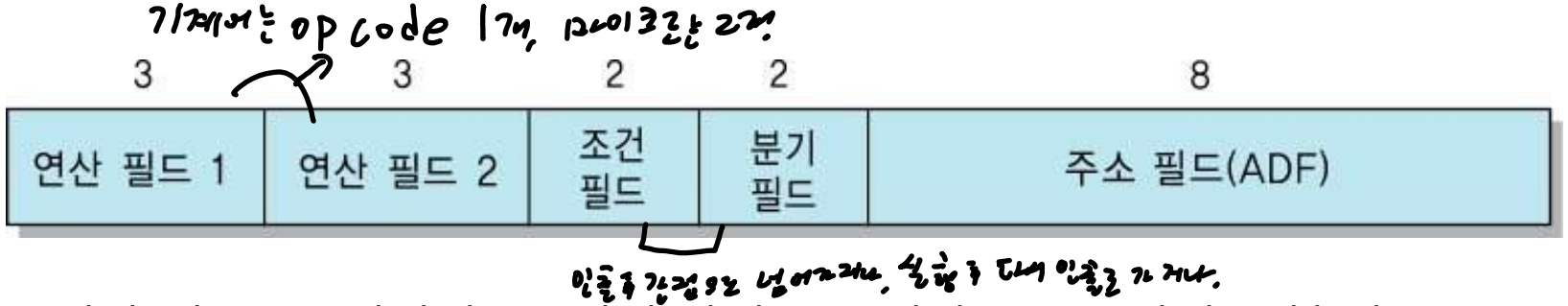
- 사상(mapping)을 이용한 해독 방법
  - 명령어의 연산 코드를 특정 비트 패턴과 조합하여
  - 실행 사이클 루틴의 시작 주소를 찾는 방법
- [예] 사상을 이용한 명령어 해독
  - 16-비트 길이의 명령어
  - 4 비트 연산 코드, 1 비트 간접 주소지정(I) 비트, 7 비트 주소



- 연산 코드 = 0001 → 실행 사이클 루틴의 시작 주소 = 1000100 (68번지)
- 연산 코드 = 0110 → 실행 사이클 루틴의 시작 주소 = 1011000 (88번지)

4.3 마이크로명령어 형식

- 마이크로 명령어의 저장 위치: 제어 기억장치
- 마이크로 명령어 형식의 예 (18비트: 제어기억장치 용량 128x18비트)



  - 연산 필드: 두 개이면, 두 개의 마이크로-연산들을 동시에 수행 가능
  - 조건(CD) 필드: 분기에 사용될 조건 플래그를 지정
  - 분기(BR) 필드: 분기의 종류, 다음에 실행할 마이크로 명령어의 주소를 결정하는 방법을 명시
  - 주소 필드(ADF): 분기가 발생하는 경우의 목적지 마이크로 명령어 주소로 사용

마이크로연산들에 대한 2진 코드 및 기호 [예]

- 연산필드 1의 마이크로-연산

코드	마이크로-연산	기 호
000	None	NOP
001	MAR ← PC	PCTAR
010	MAR ← IR(*addr*)	IRTAR
011	AC ← AC + MBR	ADD
100	MBR ← M[MAR]	READ
101	AC ← MBR	BRTAC
110	IR ← MBR	BRTIR
111	M[MAR] ← MBR	WRITE

연산필드 2의 마이크로-연산

코드	마이크로-연산	기 호
000	None	NOP
001	PC ← PC + 1	INCPC
010	MBR ← AC	ACTBR
011	MBR ← PC	PCTBR
100	PC ← MBR	BRTPC
101	MAR ← SP	SPTAR
110	AC ← AC − MBR	SUB
111	PC ← IR(*addr*)	IRTPC

조건 필드의 코드 지정

- 조건 필드 : 2 비트, 분기(필드)의 조건으로 사용
  - U: 무조건 분기 (주소필드의 주소로 분기, 즉 주소가 CAR로 적재)
  - I: 만약 I = 1이면, 간접 사이클 루틴을 호출 (유효주소를 인출)
  - S: 누산기에 저장된 데이터의 부호가 1이면(S=1), 분기
  - Z: 누산기에 저장된 데이터가 0이면(Z=1), 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 : 2 비트, 분기 동작을 지정
  - 00: 조건부 점프(JMP)
    - 조건 필드의 조건이 만족되면,
      - ADF 필드의 내용을 CAR로 적재→ 그 주소로 분기(JUMP)
    - 만족하지 않으면, CAR←CAR+1 (다음 마이크로 명령어 실행)
    - 만약, 조건 필드가 00(U)인 경우, 무조건 점프/무조건 호출 수행
  - 01: 조건부 호출(CALL)
    - 조건 필드의 조건이 만족되면,
      - ADF 필드의 내용을 CAR로 적재, 다음 마이크로 명령어(CAR+1)를 SBR에 저장
      - → 그 주소로 분기 (CALL)
    - 만족하지 않으면, CAR←CAR+1 (다음 마이크로 명령어 실행)
  - 10(RET) : 서브루틴으로부터 복귀 (SBR에 저장된 내용을 CAR로 적재)
  - 11(MAP) : 사상 방식에 의하여 분기 목적지 주소 결정 (CAR←사상된 주소)

코드	기호	설 명
00	JMP	만약 조건 = 1이면, CAR ← ADF 만약 조건 = 0이면, CAR ← CAR + 1
01	CALL	만약 조건 = 1이면, CAR ← ADF, SBR ← CAR + 1 만약 조건 = 0이면, CAR ← CAR + 1
10	RET	CAR ← SBR (서브루틴으로부터의 복귀)
11	MAP	CAR(1) ← 1, CAR(2-5) ← IR(op), CAR(6,7) ← 0

4.4 마이크로프로그래밍

- 인출 사이클 루틴
  - 인출사이클의 마이크로 명령어 루틴(인출사이클의 제어기억장치 주소: 0)

```
        ORG 0
FETCH:  PCTAR          U  JMP  NEXT   ; MAR ← PC, 다음 마이크로명령어 실행
        READ, INCPC    U  JMP  NEXT   ; BR ← M[MAR], PC = PC + 1,
                                        다음 마이크로명령어 실행.
        BRTIR          U  MAP         ; IR ← MBR, 해당 실행 사이클 루틴으로 분기.
```

  - 2진 비트 패턴

주 소	μ-ops	CD	BR	ADF
0000000	001 000	00	00	0000001
0000001	100 001	00	00	0000010
0000010	110 000	00	11	0000000

    - 주소: 각 마이크로 명령어가 저장된 제어 기억장치내의 주소
    - μ-ops: 두 개의 마이크로-연산들
    - CD: 조건 필드
    - BR: 분기 필드
    - ADF: 주소 필드

간접 사이클 루틴

- 간접 사이클의 마이크로 명령어 루틴(인출사이클 루틴 다음에 저장)
  - 간접 주소지정 방식이 사용된 경우(명령어의 I=1)
  - 기억장치에서 실제 오퍼랜드 주소를 읽음

```
        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
```

  - RET: CAR←SBR, SBR에 저장된 주소는? 답) 실행 사이클 루틴 주소 (???)
- 2진 비트 패턴

주 소	μ-ops	CD	BR	ADF
0000100	010 000	00	00	0000101
0000101	100 000	00	00	0000110
0000110	110 000	00	10	0000000

실행 사이클 루틴

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

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

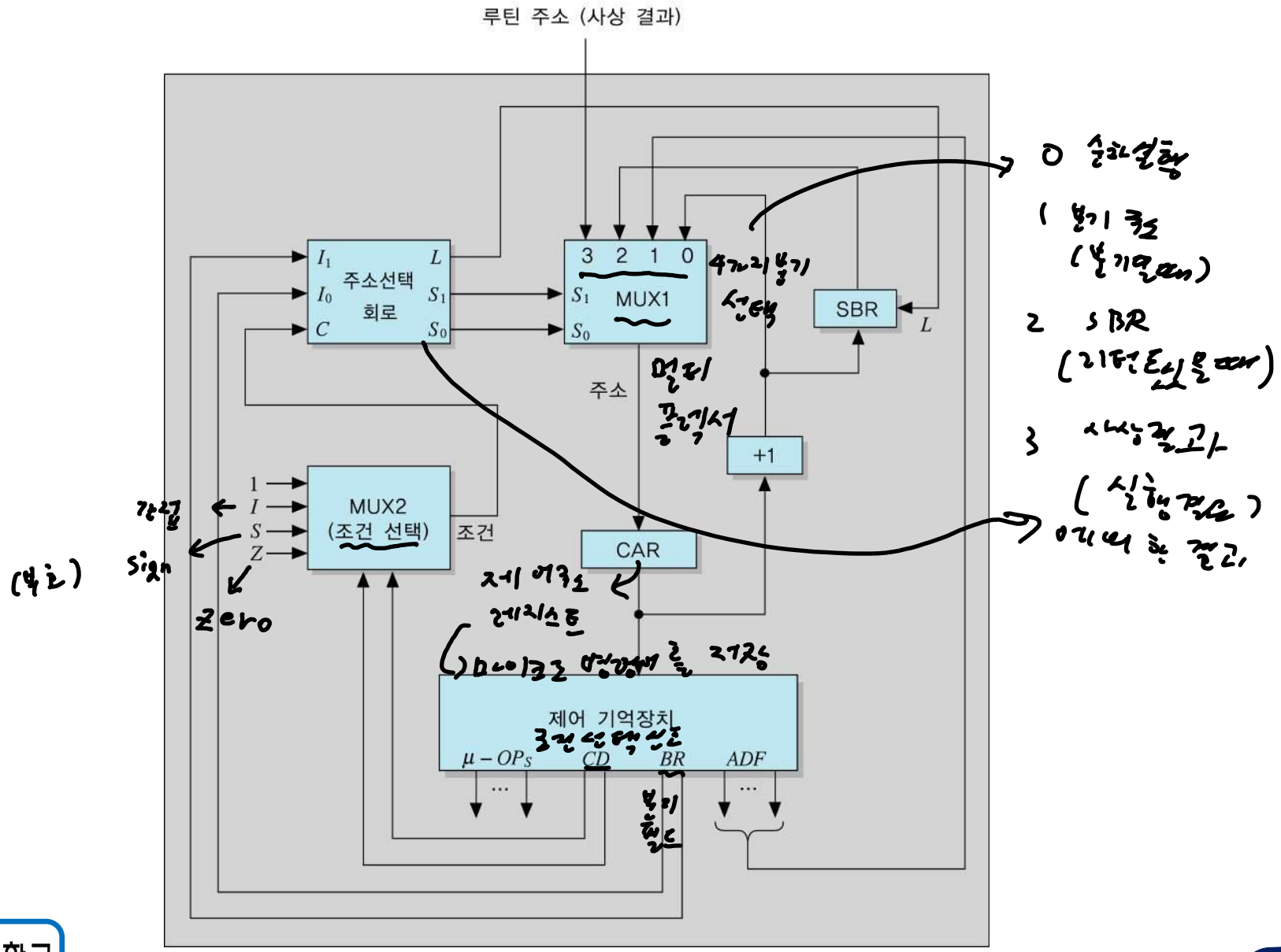
인출로 Jump.

```
        ORG 64
NOP :   INCPC    U    JMP    FETCH    ; PC ← PC+1

        ORG 68
LOAD :  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

        ORG 72
STORE : 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

        ORG 76
ADD :   IRTAR    U    JMP    NEXT     ; MAR ← IR(addr)
        READ     U    JMP    NEXT     ; MBR ← M[MAR]
        ADD      U    JMP    FETCH    ; AC ← AC + MBR

        ORG 80
SUB :   IRTAR    U    JMP    NEXT     ; MAR ← IR(addr)
        READ     U    JMP    NEXT     ; MBR ← M[MAR]
        SUB      U    JMP    FETCH    ; AC ← AC−MBR

        ORG 84
JUMP :  IRTPC    U    JMP    FETCH    ; PC ← IR(addr)
```

4.5 마이크로 프로그램의 순서제어

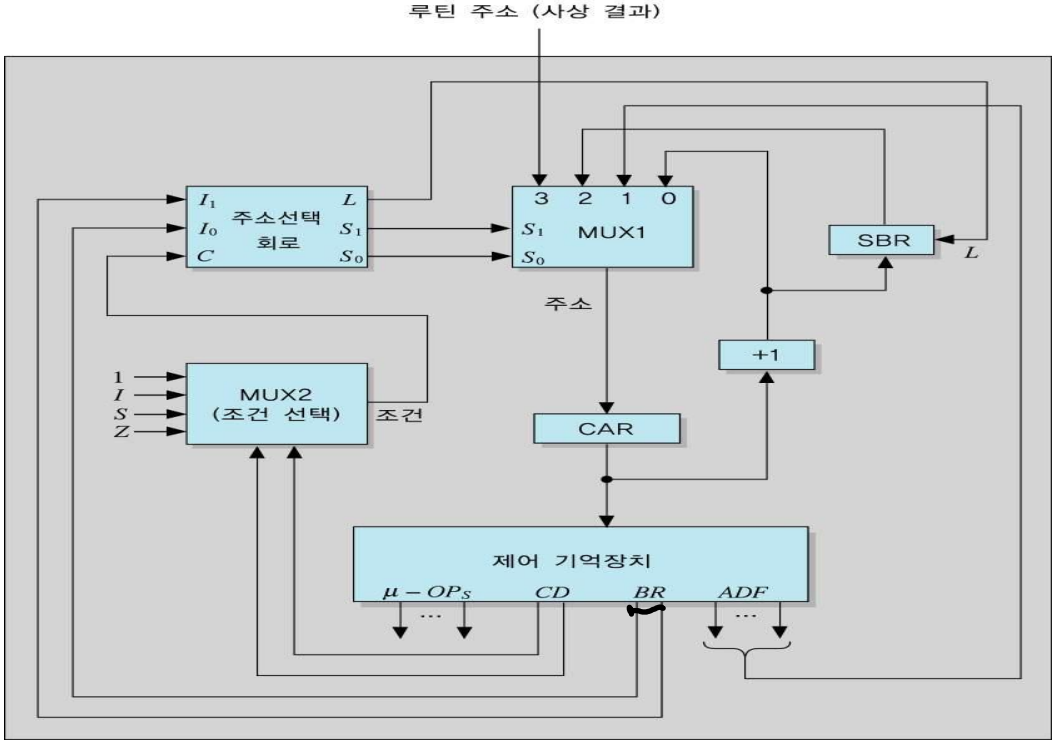
- 제어 유니트의 명령어 실행 제어
  - 제어 기억장치에 저장된 해당 마이크로 명령어들을 순서대로 인출하는 것
  - 제어 유니트의 출력 (제어 신호): 제어 기억장치로부터 읽혀진 마이크로 명령어의 연산필드의 비트가 출력됨 →제어신호
- 순서제어(sequencing)
  - 다음에 실행할 마이크로 명령어의 주소 결정
  - CAR의 값을 결정하는 것
- 제어유니트의 동작
  - CAR의 초기값 = 0 →인출 사이클 부터 시작
    - 인출 사이클 루틴의 첫 번째 마이크로 명령어의 주소
  - MUX1 : 다음에 실행할 마이크로 명령어의 주소 선택 → CAR에 저장
  - MUX2 : 조건 플래그를 선택하여 주소선택 회로로 전송

순서제어 회로가 포함된 제어 유니트의 구성도



주소 선택 방법

- 주소 선택 방법 (@주소 선택 회로, 입력: C, I0, I1, 출력: L, S0, S1)
  - BR = 00 (JUMP) 혹은 01 (CALL)일 때,
    - C = 0, 다음 위치의 마이크로 명령어 선택 (CAR←CAR+1)
    - C = 1, 주소 필드(ADF)가 지정하는 위치로 점프(jump) 혹은 호출(call)
      단, 호출 시에는 CAR 내용을 SBR에 저장 (돌아오기 때문에, 저장해 놓는 것임)
  - BR = 10 (RET)일 때: CAR←SBR (복귀)
  - BR = 11 (MAP)일 때: CAR←1XXXX00 (사상 결과를 CAR에 적재)



주소 선택 회로의 입력 및 출력 신호들

- 주소 선택회로의 동작 (MUX2, 주소선택회로, MUX1, SBR, CAR)
  - 입력: BR (분기 필드), C(조건, =CD(조건필드)의 선택 결과)
  - 출력: CAR에 저장되는 주소 (MUX1의 선택에 따라서)

BR		조건	MUX1 선택		SBR	CAR에 적재되는	설 명
I_1	I_0	C	S_1	S_0	L	MUX1의 입력	
0	0	0	0	0	0	0	CAR ← CAR + 1
0	0	1	0	1	0	1	CAR ← ADF <Jump>
0	1	0	0	0	0	0	CAR ← CAR + 1
0	1	1	0	1	1	1	SBR ← CAR, CAR ← ADF <Call>
1	0	x	1	0	0	2	CAR ← SBR <Return>
1	1	x	1	1	0	3	CAR ← 1XXXX00 <Mapping>

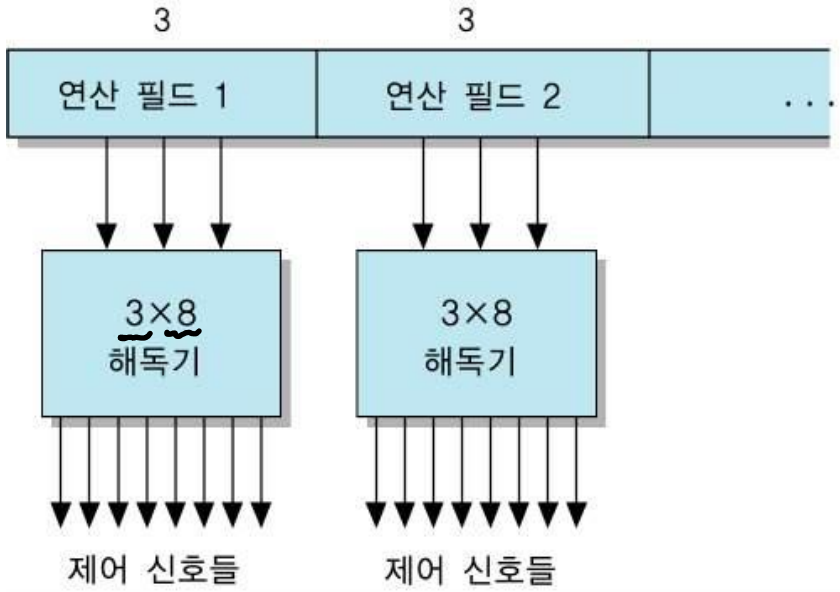
- 제어 기억장치로부터 읽혀진 마이크로 명령어의 연산 필드의 비트들
  - 제어 유니트의 외부로 나가서 제어 신호들이 된다
  - 연산 필드의 비트가 각각 3비트→6개의 제어신호 (수평적 마이크로명령어)

수평적 마이크로명령어

- Horizontal microprogramming
- 연산 필드의 각 비트와 제어 신호를 일대일로 대응
  - 그 수만큼의 비트들로 이루어진 마이크로 명령어들을 사용하는 방식
  - 해독기 없음
  - [장점] 하드웨어가 간단하고, 해독에 따른 지연 시간이 없음
  - [단점] 마이크로 명령어 비트 수가 길다.
    - 더 큰 용량의 제어 기억장치가 필요
    - (연산 필드의 비트 수가 필요한 제어 신호들의 수만큼 되어야 하기 때문)

수직적 마이크로프로그래밍

- Vertical microprogramming
- 마이크로 명령어의 연산 필드: 적은 수의 코드화된(encoded) 비트
  - 제어 기억장치의 용량을 줄이고,
  - 해독기(decoder)를 사용하여 그 코드를 필요한 수 만큼의 제어 신호들로 확장하는 방식
  - [장점] 마이크로 명령어의 비트 수가 감소
  - [단점] 해독 시간만큼의 지연 시간이 발생



Assignment #3

- 연습문제 4.1
- 연습문제 4.4

- 제출기한
  - 11월 8일 화요일