컴퓨터 구조

4장 제어 유니트1

안형태

anten@kumoh.ac.kr

디지털관 139호

제어 유니트

학습목표

- CPU의 주요 구성요소인 제어 유니트의 내부 구조와 동작 원리 분석
- 마이크로프래그래밍 기법을 이용한 명령어 세트의 설계 방법 이해

학습내용

- 제어 유니트의 기능 및 구조
- 마이크로명령어의 형식
- 마이크로프로그래밍
- 마이크로프로그램의 순서제어

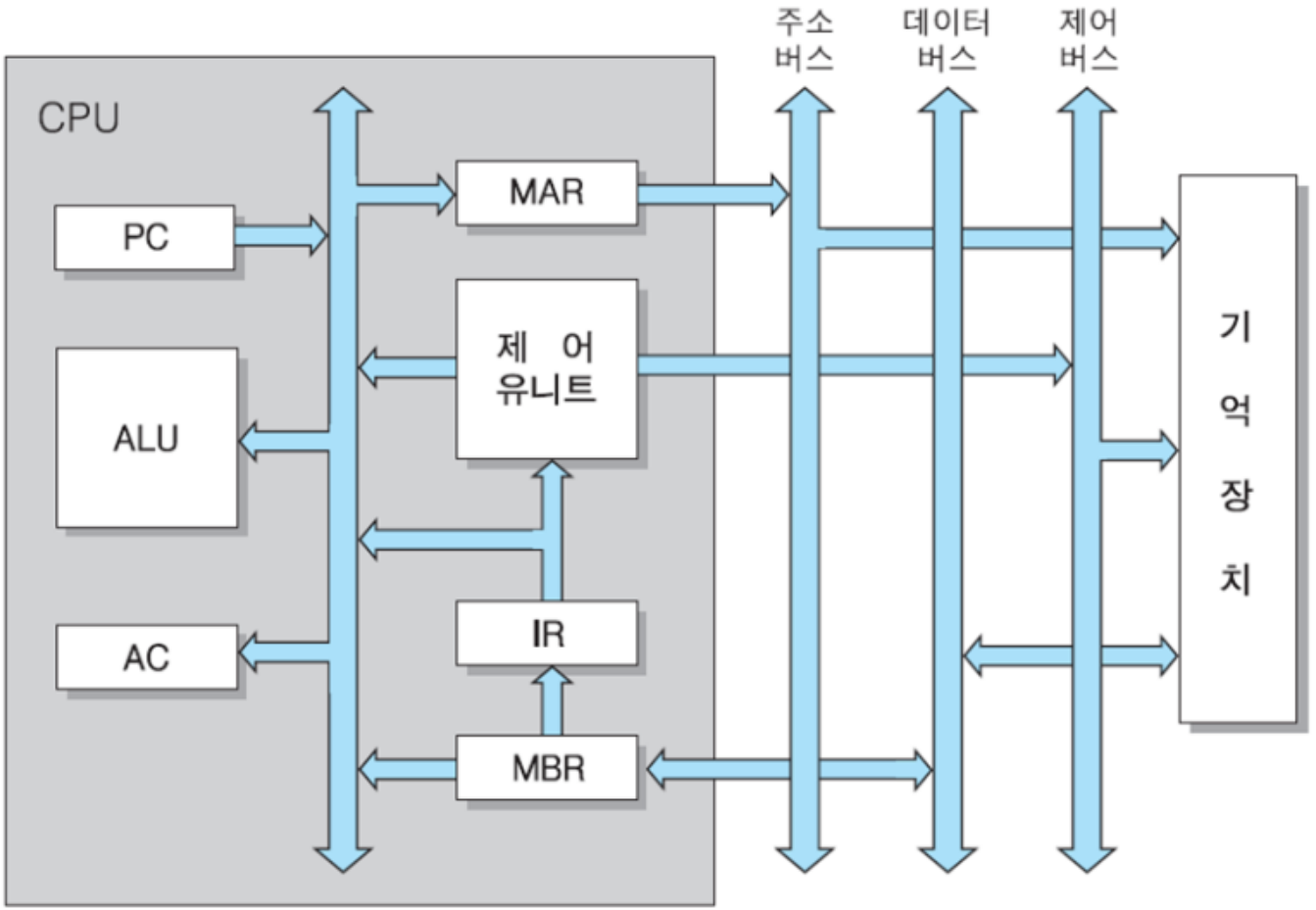
제어 유니트

1. 제어 유니트의 기능 및 구조

제어 유니트

제어 유니트(Control Unit, CU)

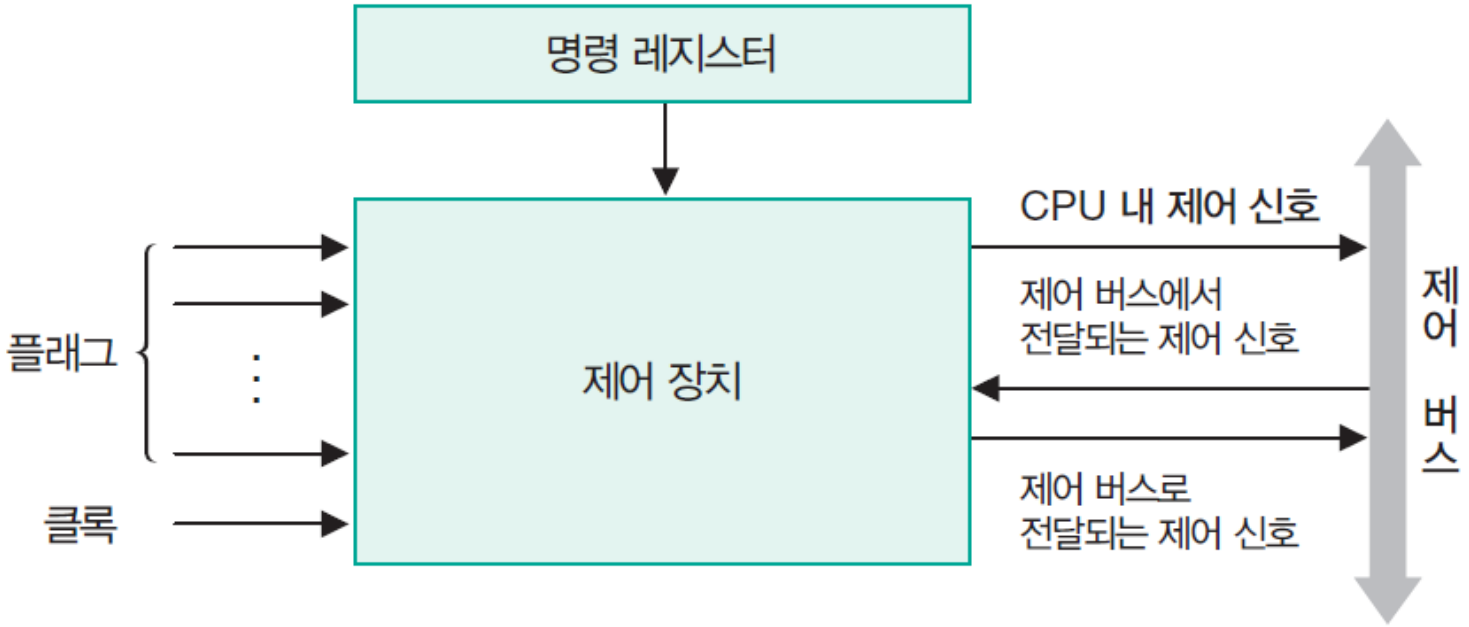
- 컴퓨터의 모든 장치의 동작을 제어하는 CPU의 핵심 장치



제어 유니트의 기능

❑ 제어 유니트의 기능

- 프로그램의 명령어를 해독
  - 주기억장치에 저장된 명령어를 해독
- 명령어 실행에 필요한 제어 신호들의 발생
  - 해독한 명령이 지시하는 연산이 수행되도록, 해당 장치에 제어 신호를 전달



제어 유니트 설계

- 하드와이어 방식의 제어 장치(hardwired control)
  - **논리 회로**인 논리 게이트와 플립플롭으로 제어 장치를 설계하여 명령어 실행 제어에 필요한 제어 신호 발생

- 마이크로프로그램 제어 장치(microprogrammed control)
  - 프로그램 내장형 컴퓨터(stored-program computer)
    - 제어용 축소된 컴퓨터(miniature computer)
  - 마이크로명령어를 **제어 기억 장치**(control memory)에 저장하고 이것을 실행하여 제어 신호를 발생
    - 명령어에 해당되는 각각의 마이크로-연산을 제어 기억 장치의 할당된 주소에 한 개씩 microcode(비트 패턴)로 작성

하드와이어 제어와 마이크로프로그램 제어 비교

하드와이어 제어	마이크로 프로그램 제어
회로 기반 기술을 사용한다.	소프트웨어 기반 기술을 사용한다.
플립플롭, 게이트, 디코더 등을 사용하여 구현한다.	마이크로 명령이 명령의 실행을 제어하는 신호를 발생시킨다.
고정 명령 형식이다.	가변 명령 형식이다(명령당 16~64비트).
ROM이 사용되지 않는다.	ROM이 사용된다.
RISC에서 주로 사용된다.	CISC에서 주로 사용된다.
해독이 빠르다.	해독이 느리다.
변경이 어렵다.	변경이 쉽다.
칩 영역이 작다.	칩 영역이 크다.

제어 유니트의 마이크로명령어

❑ 마이크로명령어(micro-instruction)

- 명령어 사이클의 각 주기에서 실행되는 각 마이크로-연산을 지정해 주는 2진 비트 패턴
  - 제어 단어(control word)라고도 함
  - 다음에 수행될 또 다른 마이크로명령어를 결정

❑ 마이크로프로그램(microprogram)

- 마이크로명령어들의 집합

❑ 루틴(routine)

- CPU의 특정 기능을 수행하기 위한 마이크로명령어들의 그룹
  - **[예]** 인출 사이클 루틴, 간접 사이클 루틴, 인터럽트 사이클 루틴, 실행 사이클 루틴

마이크로프로그램 제어 유니트의 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 마이크로명령어의 주소를 저장하는 레지스터
  - 주소는 제어 기억장치의 특정 위치를 지정

- **제어 기억장치**(control memory): 마이크로명령어들로 이루어진 마이크로프로그램을 저장하는 내부 기억장치
  - 읽기 전용 기억장치(Read-Only Memory, **ROM**)으로 구성
  - ROM의 사이즈는 마이크로명령어 형식에 따라 마이크로프로그램 크기로 결정

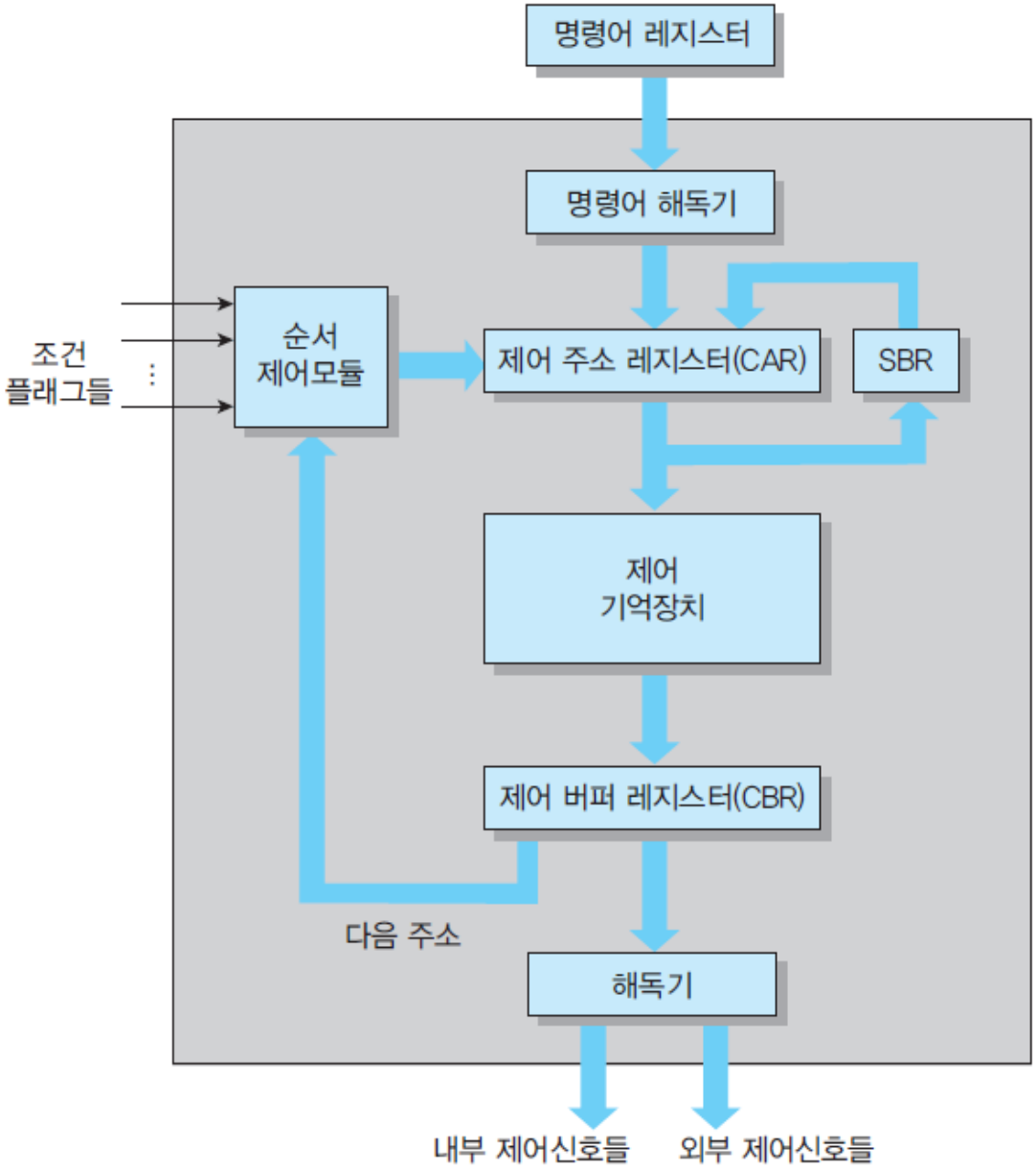
제어 유니트의 구조

- **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

제어 유니트의 내부 구성도

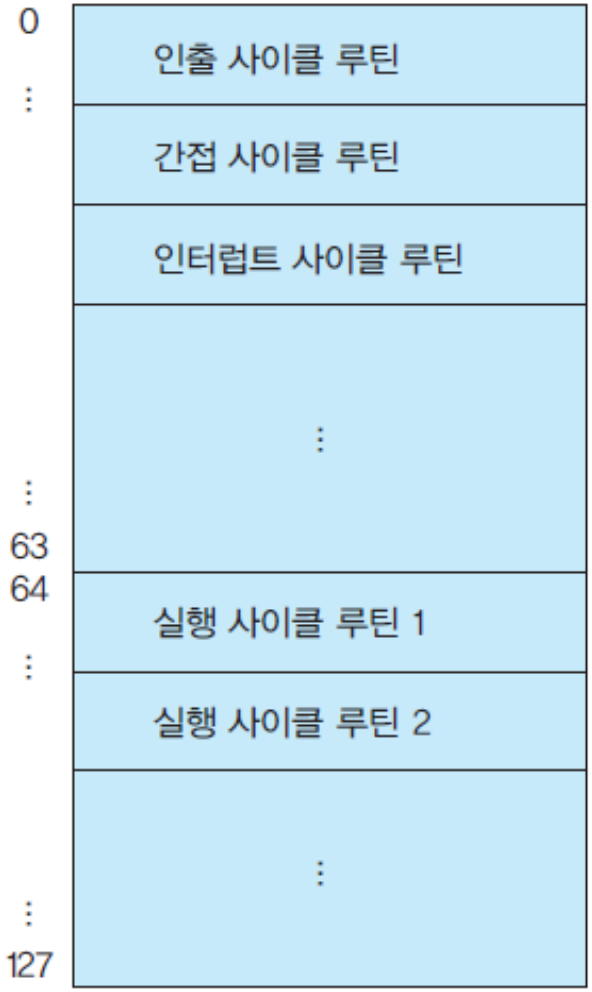


CPU의 명령어 세트 설계 및 구현 과정

- ① CPU에서 사용될 명령어의 종류 및 비트 패턴 정의
- ② 명령어들의 실행에 필요한 하드웨어 설계
- ③ 각 명령어를 위한 다양한 실행 사이클 루틴 작성
  - 마이크로프로그래밍
    - 인출 사이클, 간접 사이클, 인터럽트 사이클은 공통 루틴
- ④ 작성된 마이크로프로그램 코드들을 제어 기억장치(ROM)에 저장

제어 기억장치의 내부 구성

- **[예]** 제어 기억장치에 저장된 마이크로프로그램 루틴들
  - 제어 기억장치 용량 = 128 단어
  - 전반부(0 ~ 63번지): 공통 루틴들 저장
  - 후반부(64 ~ 127번지): 각 명령어의 실행 사이클 루틴들 저장
  - **[예]** 마이크로명령어 길이: 17 비트
    - 주소필드(Address field, **ADF**) 비트: 7 비트
    - 제어 기억장치 크기: $2^7 \times 17$ bits



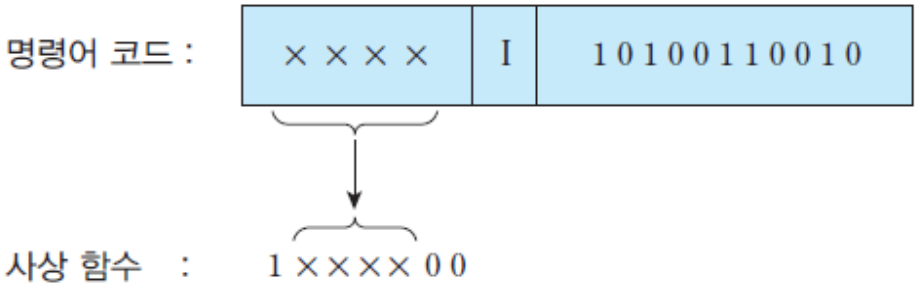
명령어 해독 과정

❑ 명령어 해독

- 명령어의 연산 코드가 지정하는 연산을 위한 실행 사이클 루틴의 시작 주소를 결정하는 동작

❑ 사상(mapping)을 이용한 해독 방법

- 명령어의 연산 코드와 특정 비트 패턴의 조합으로 실행 사이클 루틴의 시작 주소를 결정
- **[예]** 실행 사이클 루틴들이 제어 기억장치의 64번지부터 저장되어 있고, 각 루틴은 최대 네 개씩의 마이크로명령어들로 구성 된다고 가정



- 연산 코드 = 0001 → 실행 사이클 루틴의 시작 주소 = 1000100 (68_{10})
- 연산 코드 = 0110 → 실행 사이클 루틴의 시작 주소 = 1011000 (88_{10})

제어 유니트

2. 마이크로명령어의 형식

마이크로명령어의 형식

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

마이크로연산들에 대한 2진 코드 및 기호

- **[예]** '연산필드 1'에 위치할 마이크로-연산들

코드	마이크로-연산	기호
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

마이크로연산들에 대한 2진 코드 및 기호

- **[예]** '연산필드 2'에 위치할 마이크로-연산들
  - None 마이크로-연산을 제외한 '연산필드 1'과 '연산필드 2'에서 정의된 마이크로-연산들 서로 다름

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 $I = 1$이면, 간접 사이클 루틴을 호출
  - ***S:*** 누산기에 저장된 데이터의 부호가 1이면, 분기
  - ***Z:*** 누산기에 저장된 데이터가 0(Z=1)이면, 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
  - JUMP 및 CALL: 조건 필드의 조건이 만족되면, ADF 필드의 내용을 CAR로 적재 → 그 주소로 분기
    - CALL에 의해서 호출되는 루틴의 마지막 마이크로명령어의 분기 필드에는 반드시 RET로 설정되어야 됨
  - 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 ← ADF 만약 조건 = 0이면, CAR ← CAR + 1
01	CALL	만약 조건 = 1이면, SBR ← CAR + 1, CAR ← ADF 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

Q & A

End!