



8. 중앙처리장치, CPU

목표

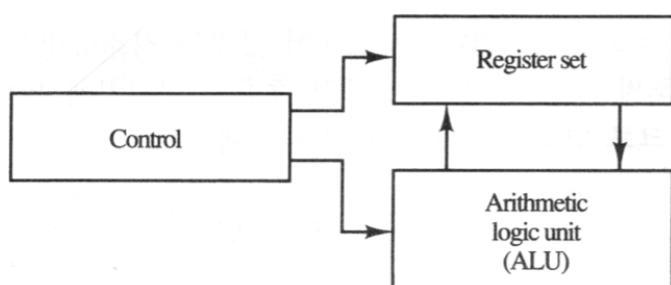
- 중앙처리장치의 기본 구조 및 명령어 형식을 이해한다.



개요

중앙처리장치

- Central Processing Unit
- 데이터 처리를 수행하는 장치



컴퓨터 구조

- 레지스터 구성
- 명령어 집합
 - 명령어 형식
 - 어드레싱 모드



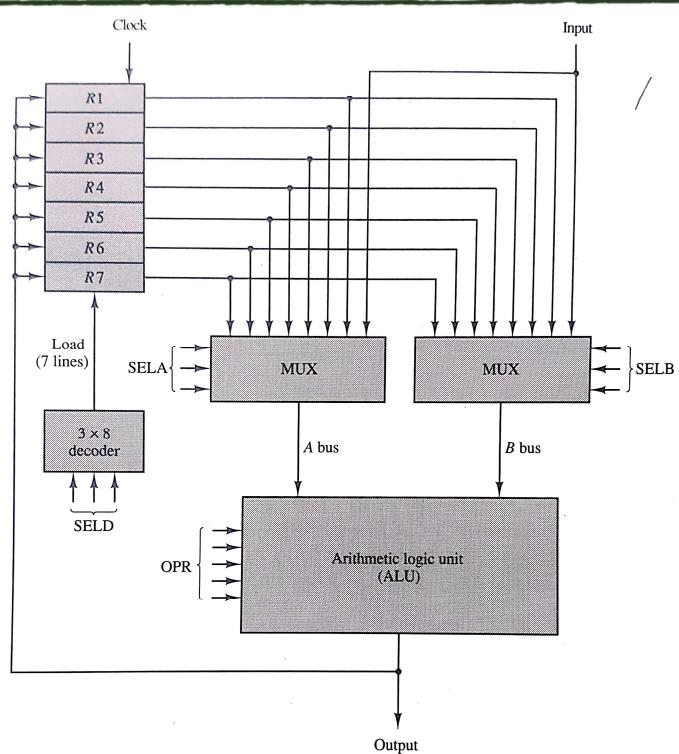
범용 레지스터 구조



범용 레지스터

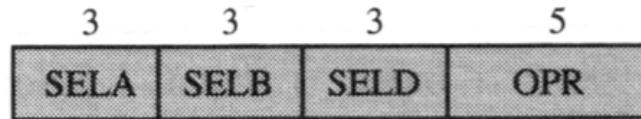
- 데이터 임시 저장
 - 포인터, 카운터, 자료, 리턴 주소, 중간 결과…
 - 메모리 저장 대비 속도 빠름
- 다수의 레지스터 연결
 - 로컬 버스
 - 빠른 접근 시간을 갖는 로컬 메모리로 이용

범용 레지스터 연결 - 버스



제어 워드

- 마이크로 연산 수행을 위한 각 제어점의 값
- 예 :



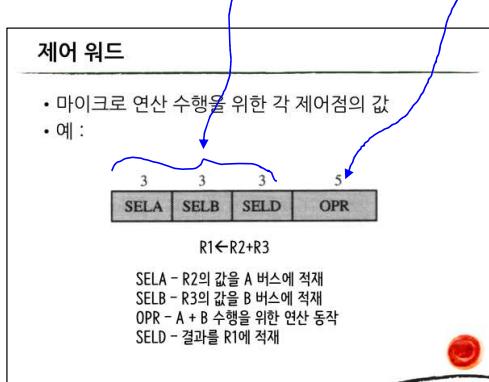
$$R1 \leftarrow R2 + R3$$

SELA - R2의 값을 A 버스에 적재
SELB - R3의 값을 B 버스에 적재
OPR - A + B 수행을 위한 연산 동작
SELD - 결과를 R1에 적재

제어워드 레지스터 선택 및 ALU 연산

Binary Code	SELA	SELB	SELD
000	Input	Input	None
001	R1	R1	R1
010	R2	R2	R2
011	R3	R3	R3
100	R4	R4	R4
101	R5	R5	R5
110	R6	R6	R6
111	R7	R7	R7

OPR Select	Operation	Symbol
00000	Transfer A	TSFA
00001	Increment A	INCA
00010	Add A + B	ADD
00101	Subtract A - B	SUB
00110	Decrement A	DECA
01000	AND A and B	AND
01010	OR A and B	OR
01100	XOR A and B	XOR
01110	Complement A	COMA
10000	Shift right A	SHRA
11000	Shift left A	SHLA



스택 구조



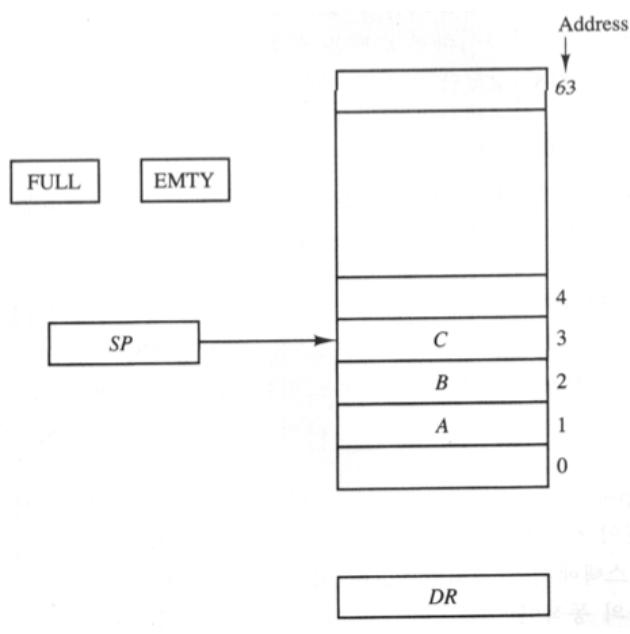
스택, stack

- Last-In-First-Out, LIFO
- SP, Stack Pointer
- 연산
 - PUSH
 - POP



스택의 구현

- 메모리의 일부분
- 제한된 수의 메모리 워드
- 레지스터



스택 PUSH

- 스택에 데이터 삽입
- 마이크로 연산 :
 $SP \leftarrow SP + 1$
 $M[SP] \leftarrow DR$
 $\text{if}(SP=0) \text{ then } (\text{FULL} \leftarrow 1)$
 $\text{EMTY} \leftarrow 0$

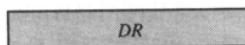
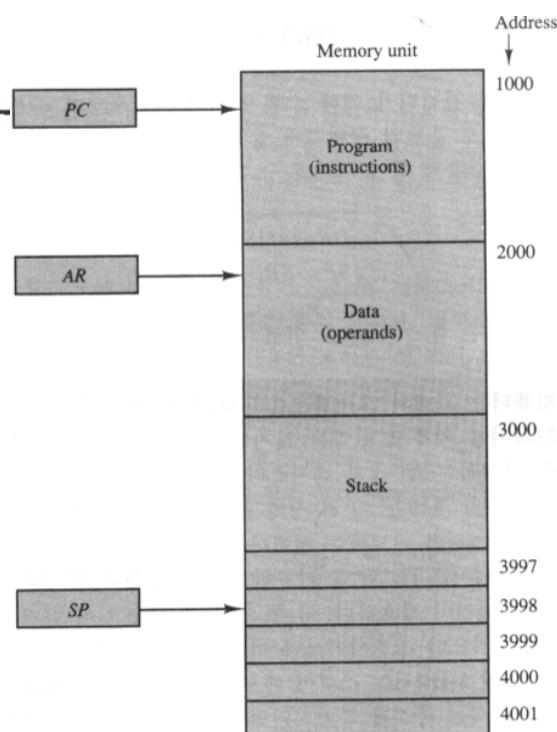
스택 POP

- 스택에 데이터 접근 & 삭제
- 마이크로 연산 :
 $DR \leftarrow M[SP]$
 $SP \leftarrow SP - 1$
 $\text{if}(SP=0) \text{ then } (\text{EMPTY} \leftarrow 1)$
 $\text{FULL} \leftarrow 0$



메모리 스택

- 주기억장치 일부 영역 이용



스택 사용

- 서브루틴/함수 호출 관련
 - 반환 주소, 매개변수
 - 루틴의 지역 변수
- 인터럽트 관련
 - 상태 저장
 - 반환 주소
- 역 Polish 표기의 연산 수행



역 Polish 표기

- 피연산자 뒤에 연산자
- Post-fix 표기법
 - (In-fix) $A * B + C * D \rightarrow$ (post-fix) A B * C D * +
- 특징
 - 연산 우선 순위 없음
 - 효율적 연산 처리/간단



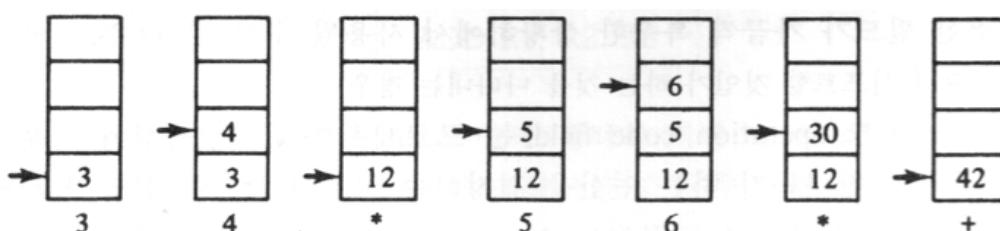
역 Polish 표기 연산법

- 왼쪽에서 오른쪽 방향으로 식 스캔
- 피연산자 - 스택 PUSH
- 연산자 - POP 2회/팝된 데이터 이용 연산 수행/연산결과는 PUSH



역 Polish 표기 연산법

3 4 * 5 6 * + 연산 수행



명령어 형식



명령어 형식

- 연산 코드 필드, Operation Code, OP-code
- 주소 필드
- 주소 모드 필드, Addressing Mode



CPU 구조

- 단일 누산기 구조
- 범용 레지스터 구조
- 스택 구조



단일 누산기 구조

- 누산기 중심의 연산 수행
- 하나의 주소 필드 - 예) ADD X



범용 레지스터 구조

- CPU내에 다수의 레지스터
- 두 개/세 개의 주소 필드 - 예) ADD R1, R2, R3



스택 구조

- 스택 이용 연산
- 연산 명령어 - 0-주소 명령어 : ADD
- 메모리 접근 명령어 - PUSH, POP



$X=(A+B)*(C+D)$ 연산 수행 예

- 3-주소 명령어

```
ADD R1, A, B  
ADD R2, C, D  
MUL X, R1, R2
```

- 1-주소 명령어

```
LOAD   A  
ADD    B  
STORE  T  
LOAD   C  
ADD    D  
MUL    T  
STORE  X
```

$X=(A+B)*(C+D)$ 연산 수행 예

- 0-주소 명령어

```
PUSH   A  
PUSH   B  
ADD  
PUSH   C  
PUSH   D  
ADD  
MUL  
POP    X
```

어드레싱 모드



주소 모드

- Addressing Mode
- 명령어 주소 필드 해석 방법
- 종류：
 - 내장모드, 즉치 모드, 레지스터 모드, …



왜 주소 모드 인가?

- 포인터, 카운터 인덱싱, 프로그램 재배치 등의 편의제공
- 명령어의 주소 필드의 비트 감소



주소 모드 종류

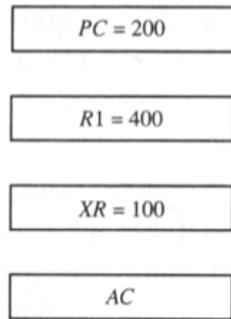
- | | |
|----------------|--------------------|
| • implied 모드 | • 직접주소 |
| • immediate 모드 | • 간접주소 |
| • 레지스터 모드 | • 상대주소 |
| • 레지스터 간접 모드 | • 인덱스드 어드레싱 모드 |
| • 자동증가/자동감소 | • 베이스 레지스터 어드레싱 모드 |



주소 모드 예

※ 한 명령어(instruction)는 두 개의 메모리 워드에 저장됨을 가정.

Addressing Mode	Effective Address	Content of AC
Direct address	500	800
Immediate operand	201	500
Indirect address	800	300
Relative address	702	325
Indexed address	600	900
Register	—	400
Register indirect	400	700
Autoincrement	400	700
Autodecrement	399	450



Address	Memory
200	Load to AC Mode
201	Address = 500
202	Next instruction
399	450
400	700
500	800
600	900
702	325
800	300



데이터 전송과 처리



명령어 종류

- 데이터 전송 명령어
- 데이터 처리 명령어
- 프로그램 제어 명령어



데이터 전송 명령어

- 메모리 - 레지스터
- 레지스터 - 레지스터
- 레지스터 - 입출력 장치

Name	Mnemonic
Load	LD
Store	ST
Move	MOV
Exchange	XCH
Input	IN
Output	OUT
Push	PUSH
Pop	POP



데이터 처리 명령어 - 산술명령어

Name	Mnemonic
Increment	INC
Decrement	DEC
Add	ADD
Subtract	SUB
Multiply	MUL
Divide	DIV
Add with carry	ADDC
Subtract with borrow	SUBB
Negate(2's complement)	NEG



데이터 처리 명령어 - 논리명령어/비트처리 명령어

Name	Mnemonic
Clear	CLR
Complement	COM
AND	AND
OR	OR
Exclusive-OR	XOR
Clear carry	CLRC
Set carry	SETC
Complement carry	COMC
Enable interrupt	EI
Disable interrupt	DI



데이터 처리 명령어 - 시프트 명령어

Name	Mnemonic
Logical shift right	SHR
Logical shift left	SHL
Arithmetic shift right	SHRA
Arithmetic shift left	SHLA
Rotate right	ROR
Rotate left	ROL
Rotate right through carry	RORC
Rotate left through carry	ROLC



프로그램 제어 명령어

- 명령어 수행 순서 변경
- 무조건 분기
- 조건 분기

Name	Mnemonic
Branch	BR
Jump	JMP
Skip	SKP
Call	CALL
Return	RET
Compare (by subtraction)	CMP
Test (by ANDing)	TST



프로그램 제어



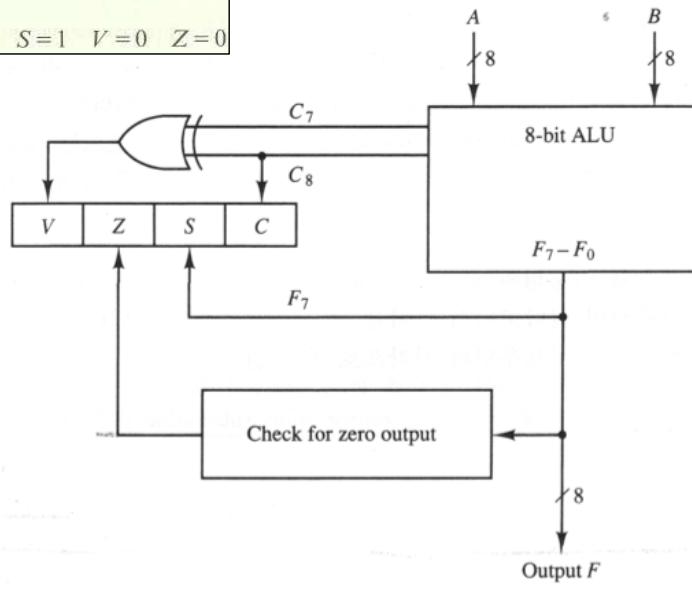
상태 비트

- ALU 연산 수행 후 상태
- 상태 레지스터 저장
 - PSW - Program Status Word
- 조건 분기문의 조건



상태 비트

$$\begin{array}{l} A: \quad 11110000 \\ \bar{B} + 1: \quad + 11101100 \\ \hline A - B: \quad 11011100 \quad C = 1 \quad S = 1 \quad V = 0 \quad Z = 0 \end{array}$$



서브루틴 call & return

- CALL 처리
 - 반환 주소 저장 - PC 저장 // **스택** 이용
 - 서브루틴 시작 주소로 제어 이동

$$\begin{aligned} SP &\leftarrow SP+1 \\ M[SP] &\leftarrow PC \\ PC &\leftarrow \text{유효주소} \end{aligned}$$

서브루틴 call & return

- RETURN 처리
 - 저장된 반환 주소로 분기

$PC \leftarrow M[SP]$

$SP \leftarrow SP - 1$



인터럽트

- 어떤 원인에 의해 발생한 사건 처리
- 인터럽트 서비스 루틴에서 처리
- 종류
 - 외부 인터럽트
 - 내부 인터럽트
 - 소프트웨어 인터럽트

]} 하드웨어 신호에 의해 발생



인터럽트 처리 절차

- 인터럽트 발생
- 인터럽트 발생 장치 결정 - H/W적으로
- 인터럽트 서비스 루틴 결정
- 상태 저장
- 인터럽트 서비스 루틴 분기
- 반환



간소화된 명령어 집합 컴퓨터
RISC



프로세서 분류

- 명령어 집합의 크기에 따라
- CISC, Complex Instruction Set Computer
- RISC, Reduced Instruction Set Computer



CISC

- 풍부한 명령어 집합
- 다양한 어드레싱 모드
- 명령어 길이 가변
- 하드웨어 복잡도 높음/낮은 수행 성능



RISC

- 적은 수의 명령어/어드레싱 모드
- 메모리 참조는 Load/Store 명령어
- 모든 연산은 범용 레지스터 이용
- 많은 레지스터
- 명령어 길이 고정
- 높은 성능

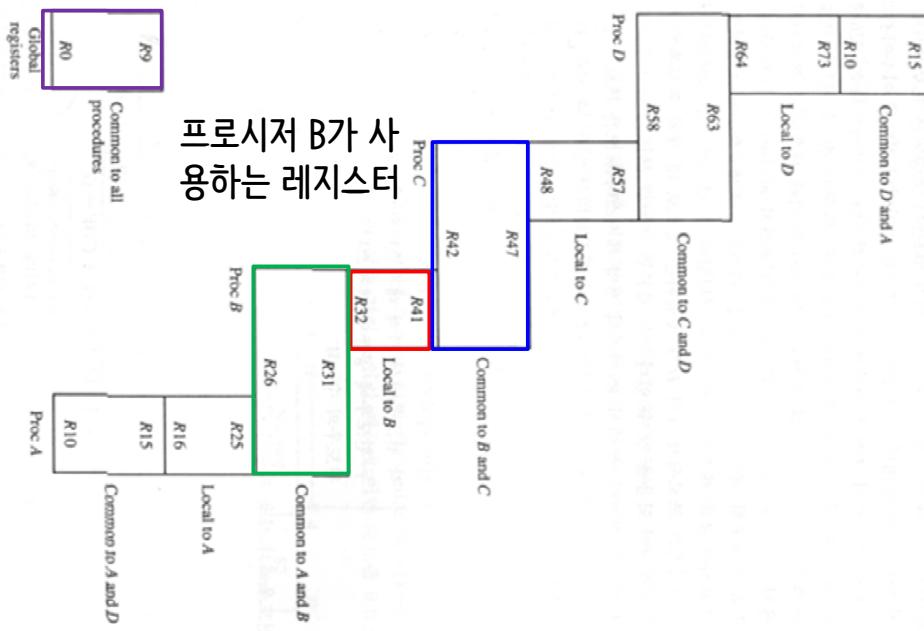


중첩된 레지스터 윈도우

- 함수 파라미터/반환값 저장 - 공통 레지스터
- 지역 변수 저장 - 지역 레지스터
- 하나의 윈도우만 활성화



중첩된 레지스터 윈도우



중첩된 레지스터 윈도우

전역 레지스터의 수 = G

각 윈도우의 지역 레지스터의 수 = L

두 윈도우가 공유하는 공통 레지스터의 수 = C

윈도우의 수 = W

윈도우 크기 = L + 2C + G

레지스터 파일 = (L + C) W + G



Q&A

