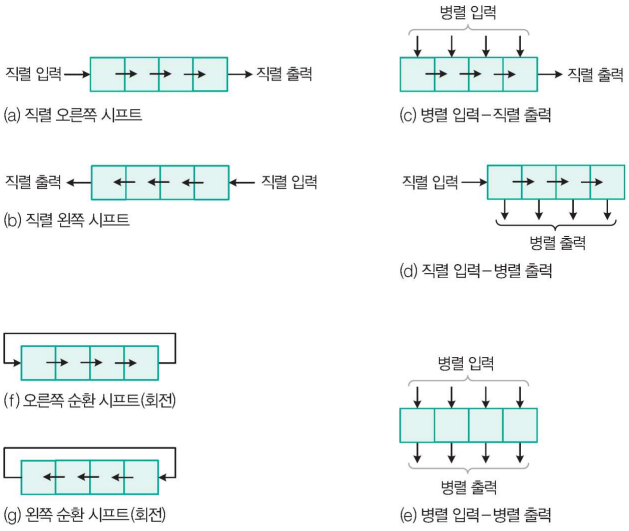
04 순서 논리 회로

5 레지스터

- **레지스터**(register)는 기본적으로 데이터 비트를 저장하는 소자
- 대부분의 레지스터에서는 D 플립플롭이 사용되며 각 D 플립플롭에 한 비트씩 저장
- 따라서 n비트 레지스터는 플립플롭 n개로 구성되며, n비트의 2진 정보를 저장할 수 있다.



03 레지스터

1 레지스터 동작

- 레지스터는 CPU가 사용하는 데이터와 명령어를 신속하게 읽어 오고 저장하고 전송하는 데 사용
- 레지스터는 메모리 계층의 최상위에 있으며 시스템에서 가장 빠른 메모리
- 매우 단순한 마이크로프로세서는 누산기(AC) 레지스터 1개로만 구성 가능
- 레지스터 용도에 따른 종류
  - 누산기(Accumulator, AC)
  - 프로그램 카운터(Program Counter, PC)
  - 명령 레지스터(Instruction Register, IR)
  - 인덱스 레지스터(Index Register, IX)
  - 스택 포인터(Stack Pointer, SP)
  - 메모리 데이터 레지스터(Memory Data Register : MDR, Memory Buffer Register : MBR)
  - 메모리 주소 레지스터(Memory Address Register, MAR)
- 데이터(범용) 레지스터는 보통 8~32개 정도, 많으면 128개 이상인 경우도 있음
- 특수 레지스터는 8~16개 정도

03 레지스터

❖ 레지스터 동작 개념

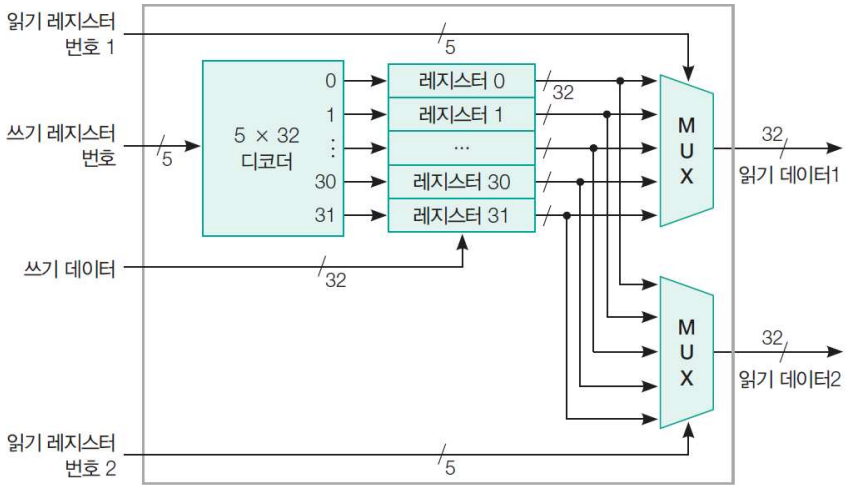


그림 4-6 레지스터 동작 개념

03 레지스터

2 레지스터 종류

❖ **메모리 주소 레지스터**(Memory Address Register, MAR)

- CPU가 읽고 쓰기 위한 데이터의 메모리 주소 저장
- 메모리에 데이터를 저장하거나 읽을 때 필요한 메모리 위치의 주소를 MAR로 전송

❖ **메모리 버퍼 레지스터**(Memory Buffer Register, MBR, MDR)

- 메모리에서 데이터를 읽거나 메모리에 저장될 명령의 데이터를 일시적 저장
- 명령어 내용은 명령 레지스터로 전송되고, 데이터 내용은 누산기 또는 I/O 레지스터로 전송

❖ **입출력 주소 레지스터**(I/O Address Register, I/O AR)

- 특정 I/O 장치의 주소를 지정하는 데 사용

❖ **입출력 버퍼 레지스터**(I/O Buffer Register, I/O BR)

- I/O 모듈과 프로세서 간에 데이터를 교환하는 데 사용

03 레지스터

❖ **프로그램 카운터**(PC)

- 명령 포인터 레지스터라고도 하며, 실행을 위해 인출(fetch)할 다음 명령의 주소를 저장하는데 사용
- 명령어가 인출되면 PC 값이 단위 길이(명령 크기)만큼 증가
- 항상 가져올 다음 명령의 주소 유지

❖ **명령 레지스터**(Instruction Register, IR)

- 주기억 장치에서 인출한 명령어 저장
- 제어 장치는 IR에서 명령어를 읽어 와서 해독하고 명령을 수행하기 위해 컴퓨터의 각 장치에 제어신호 전송

❖ **누산기**(ACcumulator register, AC)

- ALU 내부에 위치하며, ALU의 산술 연산과 논리 연산 과정에 사용
- 제어 장치는 주기억 장치에서 인출된 데이터 값을 산술 연산 또는 논리 연산을 위해 누산기에 저장
- 이 레지스터는 연산할 초기 데이터, 중간 결과 및 최종 연산 결과 저장
- 최종 결과는 목적지 레지스터나 MBR을 이용하여 주기억 장치로 전송

03 레지스터

❖ **스택 제어 레지스터**(Stack Control Register, Stack Pointer)

- 메모리의 한 블록이며, 데이터는 후입 선출(Last In-First Out, LIFO)로 검색
- 메모리 스택을 관리하는 데 사용
- 크기는 2 또는 4바이트

❖ **플래그 레지스터**(Flag Register, FR)

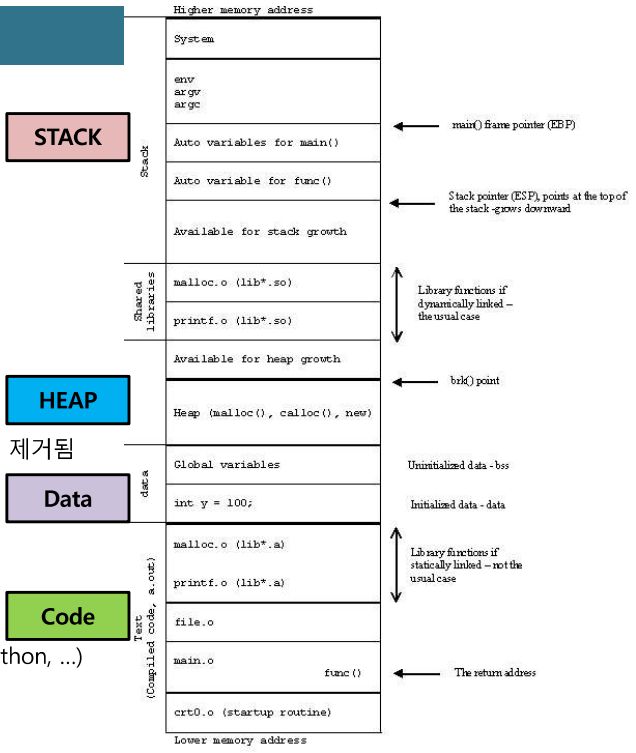
- CPU가 작동하는 동안 특정 조건의 발생을 표시하는 데 사용
- 1바이트 또는 2바이트인 특수 목적 레지스터
- 예를 들어 산술 연산 또는 비교 결과로 제로 값이 누산기에 입력되면 제로 플래그를 1로 설정
- 상태 레지스터(Status Register, SR), 프로그램 상태 워드(Program Status Word, PSW)라고도 함

❖ **데이터 레지스터**(Data Register, 범용 레지스터)

- CPU내의 데이터를 일시적으로 저장하기 위한 레지스터
- 고정 소수, 부동 소수로 구분하여 따로 저장하는 경우도 있으며,
- 어떤 프로세서는 상수 0 또는 1을 저장할 수 있도록 하는 레지스터도 있다.

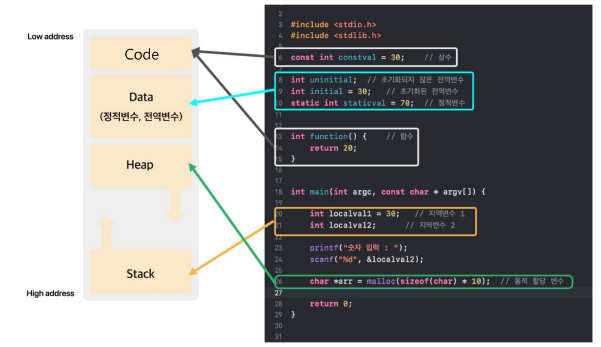
Memory Segments

- **코드(Code) 영역**
  - 실행할 프로그램의 코드가 저장되는 영역
- **데이터(Data) 영역**
  - 전역변수/정적변수/초기화된변수 저장
- **스택(Stack) 영역**
  - 함수 호출과 관계되는 지역 변수, 매개변수 저장
  - 함수 호출시 할당, 함수 종료시 소멸
  - Push/Pop 으로 동작
  - 코드 블록 { } 안에서 생성된 변수들은 블록이 닫히면 스택에서 제거됨
  - Stack Overflow (스택 메모리 부족시)
- **힙(Heap) 영역**
  - 사용자가 직접 생성(new) 및 해제(delete) 해야하는 메모리
  - 프로그램 종료되어도 해제되지 않음
  - Garbage Collection 기능이 주기적으로 제거해줌(JAVA VM, Python, ...)



Memory Segments

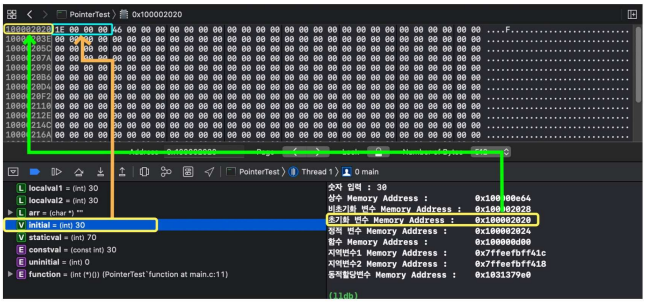
- https://st-lab.tistory.com/198



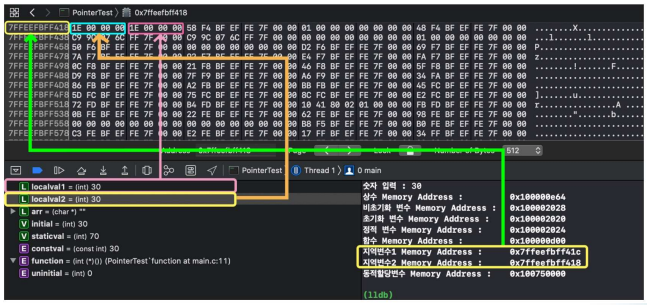
스택

```
상수 Memory Address :          0000000100000e44
비초기화 변수 Memory Address :  0000000100002028
초기화 변수 Memory Address :    0000000100002020
정적 변수 Memory Address :      0000000100002024
함수 Memory Address :           0000000100000ce0
지역변수1 Memory Address :      00007ffeefbff47c
지역변수2 Memory Address :      00007ffeefbff478
동적할당변수 Memory Address :   0000000103006680
```

[initial]



[localval1 & localval2]



03 레지스터

❖ 인텔 x86 레지스터 종류

The x86 architecture consists of the following unprivileged integer registers.

eax	Accumulator
ebx	Base register
ecx	Counter register
edx	Data register – can be used for I/O port access and arithmetic functions
esi	Source index register
edi	Destination index register
ebp	Base pointer register
esp	Stack pointer

x86 Flags

In the preceding example, the two-letter codes at the end of the second line are *flags*. These are single-bit registers and have a variety of uses.

The following table lists the x86 flags:

Flag Code	Flag Name	Value	Flag Status	Description
of	Overflow Flag	0 1	nvov	No overflow - Overflow
df	Direction Flag	0 1	updn	Direction up - Direction down
if	Interrupt Flag	0 1	dixi	Interrupts disabled - Interrupts enabled
sf	Sign Flag	0 1	ping	Positive (or zero) - Negative
zf	Zero Flag	0 1	nzzr	Nonzero - Zero
af	Auxiliary Carry Flag	0 1	naac	No auxiliary carry - Auxiliary carry
pf	Parity Flag	0 1	pepo	Parity odd - Parity even
cf	Carry Flag	0 1	nccy	No carry - Carry
tf	Trap Flag			If **tf** equals 1, the processor will raise a STATUS_SINGLE_STEP exception after the execution of one instruction. This flag is used by a debugger to implement single-step tracing. It should not be used by other applications.
iopl	I/O Privilege Level			I/O Privilege Level This is a two-bit integer, with values between zero and 3. It is used by the operating system to control access to hardware. It should not be used by applications.

그림 4-7 인텔 x86 레지스터 종류

https://learn.microsoft.com/en-us/windows-hardware/drivers/debugger/x86-architecture

03 레지스터

3 레지스터 전송(LOAD, STORE, MOVE 명령 등)

- 3가지 레지스터 전송 명령 : LOAD, STORE, MOVE

LOAD	• 주기억 장치에서 레지스터로 데이터를 읽음
STORE	• 레지스터에서 주기억 장치로 데이터를 저장
MOVE	• 레지스터에서 레지스터로 데이터를 이동

- 인텔 프로세서는 이 세 가지를 **MOVE 명령어 하나로** 모두 처리한다.
- MOVE 명령어를 사용하여 데이터 교환이나 데이터형 변환이 가능하다.

❖ 데이터 교환

- MOVE 명령을 세 번 사용하여 두 오퍼랜드를 교환할 수 있다.
- 바이트 교환을 이용하여 빅 엔디안을 리틀 엔디안으로 또는 그 반대로 만들 수 있다.

❖ 데이터형 변환

- 크기가 작은 레지스터에 저장된 정수를 큰 레지스터로 이동하여 데이터형을 변환한다.
- 8비트→16비트, 16비트→32비트, 32비트→64비트 등

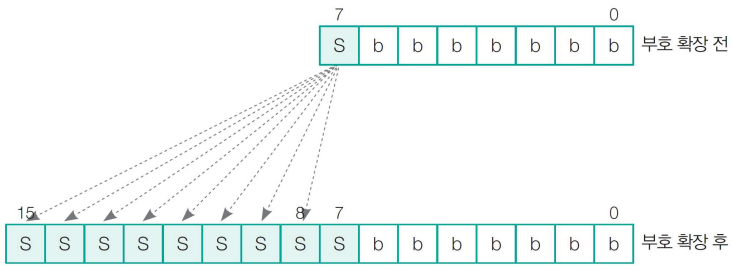


그림 4-8 부호 확장