<컴퓨터구조의 세부분야>

출처: DreamHack

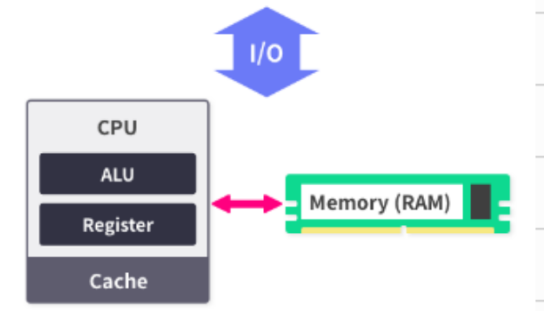
*실명출처도 DreamHack

컴퓨터 구조의 세부 분야 ✓

- 기능 구조의 설계 → 컴퓨터가 연산을 효율적으로 하기 위해 어떤 기능들이 컴퓨터에 필요할지 고민하고 설계한다
  - 폰 노이만 구조
  - 하버드 구조
  - 수정된 하버드 구조
- 명령어 집합구조 → CPU가 처리해야하는 명령어를 설계하는 분야
  - x86, x86-64
  - ARM
  - MIPS
  - AVR
- 마이크로 아키텍처 → CPU의 하드웨어적 설계 정의된 명령어집합을 효율적으로 처리할 수 있도록 CPU 회로를 설계한다
  - 캐시 설계
  - 파이프라이닝
  - 슈퍼 스칼라
  - 분기 예측
  - 비순차적 명령어 처리
- 하드웨어 및 컴퓨팅 방법론
  - 직접 메모리 접근

출처: DreamHack

폰 노이만 구조



<폰 노이만 구조>

Idea: 컴퓨터에 연산/제어/저장 3가지 핵심 기능 필요 + bus (장치간 데이터와 제어신호 교환을 위해)

연산/제어 → 중앙처리장치 CPU

저장 → memory

1) CPU (중앙처리장치)

* 하는 일: 프로세스의 코드 불러오기/실행하기
  결과를 저장하기
* 구성
  - ALU (산술, 논리 연산을 처리한다)
  - Control Unit (CPU를 제어한다)
  - 레지스터 (CPU에 필요한 데이터를 저장)

2) 기억장치

* 용도에 따른 분류
  - 주기억장치
  - 보조기억장치
* 주기억장치: 프로그램 실행과정에서 필요한 데이터를 임시로 저장한다 ex) RAM
* 보조기억장치: 운영체제, 프로그램 등과 같은 데이터를 장기간 보관할 때.
  ex) 하드 드라이브(HDD), SSD 등

3) bus (버스)

* 컴퓨터 부품 - 부품 / 컴퓨터 - 컴퓨터 사이 신호를 전송하는 통로
* Data bus: 데이터 이동
* address bus: 주소를 지정
* control bus: 읽기/쓰기를 제어함
* 이외: 랜선, 데이터전송 소프트웨어, 프로토콜

<x86-64 아키텍처> ⇒ Intel64, IA-32e, EM64T, amd64...

1) n비트 아키텍처

: n비트란 컴퓨터가 한 번에 처리할수 있는 데이터의 크기 ⇒ 32bit 아키텍처에서 ALU는 32bit까지 계산가능하다.
⇒ CPU가 이해할수 있는 데이터의 단위 ⇒ Word

* 레지스터가 한 번에 표현가능한 값의 크기: 2^{32}
  ⇒ CPU가 한 번에 인식하여 처리할수 있는 주소값의 범위
  ⇒ 32bit 운영에서 레지스터는 2^{32}개의 주소공간 이용
  ⇒ 주소공간 단위는 1byte ⇒ $2^{32} \times 1byte = 2^{32}byte = 4GB$

2) Word가 크면 유리한 점

: 워드가 클수록 제공가능한 가상메모리 크기가 ↑

→ 32bit가 제공하는 가상메모리: 4GB (2^{32}) → 64bit가 제공하는 가상메모리: $2^{64}byte$ = 16엑사바이트)

<x86-64 아키텍처: 레지스터>

* CPU가 데이터를 빠르게 저장하고 사용할 때 이용하는 보관소
* 산술연산에 필요한 데이터를 저장, 주소 저장 및 참조

* 레지스터 종류 (x64 아키텍처 기준)

1) 범용 레지스터 (주용도 有 but 자유롭게)

8바이트를 저장 가능(64bit)

⇒ 부호 없는 정수 기준 $2^{64}-1$ 수 나타냄

이름	주용도
rax (accumulator register)	함수의 반환 값
rbx (base register)	x64에서는 주된 용도 없음
rcx (counter register)	반복문의 반복 횟수, 각종 연산의 시행 횟수
rdx (data register)	x64에서는 주된 용도 없음
rsi (source index)	데이터를 옮길 때 원본을 가리키는 포인터
rdi (destination index)	데이터를 옮길 때 목적지를 가리키는 포인터
rsp (stack pointer)	사용중인 스택의 위치를 가리키는 포인터
rbp (stack base pointer)	스택의 바닥을 가리키는 포인터

2) 세그먼트 레지스터

* 종류 (메모리보호를위해존재)

→ cs, ss, ds, es, fs, gs

* 크기: 16bit (= 2byte)

* cs, ss, ds 레지스터는 코드영역, data 영역, stack memory 영역을 가리킬 때 사용

* 나머지 레지스터는 운영체제 별로 용도를 결정할수 있는 범용 레지스터

* 과거에는 메모리 세그멘테이션이나 가용 메모리 공간 확장을 위해 사용

3) 명령어 포인터 레지스터

☆ cpu가 어느부분의 코드를 실행할지 가리키는 역할 (rip 레지스터)

→ 크기: 8byte (64bit)

4) 플래그 레지스터

→ 프로세스의 현재 상태 저장

→ 크기: 64bit (8byte)

(최대 64개의 플래그 사용가능, 하지만 실제로는 20여개 사용)

플래그	의미
CF(Carry Flag)	부호 없는 수의 연산 결과가 비트의 범위를 넘을 경우 설정 됩니다.
ZF(Zero Flag)	연산의 결과가 0일 경우 설정 됩니다.
SF(Sign Flag)	연산의 결과가 음수일 경우 설정 됩니다.
OF(Overflow Flag)	부호 있는 수의 연산 결과가 비트 범위를 넘을 경우 설정 됩니다.

<레지스터 호환>

x86-64 아키텍처는 IA-32의 64bit 확장 아키텍처. ⇒ 호환이 가능하다

IA-32는 IA-16의 32bit 확장 아키텍처 ⇒ 64bit, 32bit, 16bit 호환 ○

x86-64의 레지스터