# 운영체제 1. 메모리

**CPU** : 프로그램 명령어를 실행, 프로세서라고도 함

-       제어장치 : 명령어 해석, 다른 장치들에게 동작 지시

-       연산장치 : 덧셈, 뺄셈 등의 연산

-       레지스터 : 명령어를 실행하는 동안 필요한 정보 저장

* 어드레스 레지스터(AR) : 데이터를 읽거나 쓸 메모리 주소 저장
* 프로그램 카운터(PC) : 다음에 실행될 명령어가 저장된 메모리 주소 저장
* 데이터 레지스터 : 읽어 온 데이터 저장
* 명령어 레지스터(IR) : 읽어 온(현재 실행하는) 명령어 저장
* 어큐뮬레이터(누산기) : 연산에 사용되는 데이터 저장

**메인 메모리** : 실행중인 프로그램과 프로그램에 필요한 데이터 일시적 저장

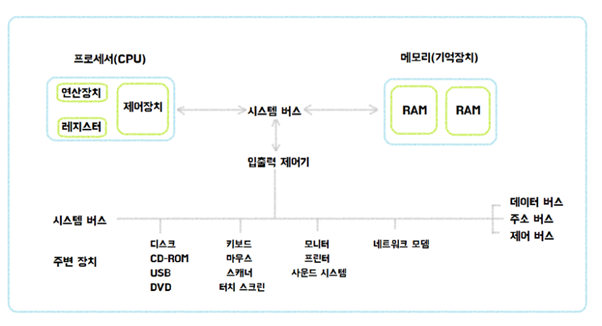
**입출력 장치** : 메인 메모리에 데이터를 입력하거나 출력 (키보드, 마우스, 모니터, 하드 디스크…)

**시스템 버스** :

-       어드레스 버스 (메모리부터 읽을 데이터가 있는 어드레스 전달)

-       데이터 버스 (메인메모리에서 읽거나 쓸, 입출력 장치에서 입력받거나 출력할 데이터 전달)

-       제어 버스 (cpu의 제어 정보 전달)



**메모리** : CPU가 프로그램을 실행하거나 데이터를 참조하기 위해 필요한 정보들을 저장해두는 공간

**메모리 계층구조** : 레지스터(cpu내부) – 캐시메모리(cpu내부) – 메인메모리 – 하드디스크

* 왼쪽으로 갈수록 CPU랑 가까워 가격 비쌈, 속도 빠름, 용량 작음

1. 보조기억장치(HDD, SSD) = 하드디스크

불필요한 프로그램이지만, 저장되어야 할 것들 (DVD, CD 등)

1. 주기억장치(RAM : Random Access Memory) = 메인메모리

당장 실행되어야 할 것들(현재 실행중인 프로그램의 데이터나 처리중인 결과) 저장

하드디스크로부터 일정량의 데이터를 복사해서 저장하고, 필요할 때마다 CPU에 전달

전원이 꺼지면 데이터 지워짐(휘발성)

* + 로딩(적재) : 하드디스크에서 데이터를 읽어 램으로 전송하는 과정

1. 캐시메모리

메인메모리(RAM)과 프로세서(CPU)의 속도차이를 보완하기 위해 존재

메모리 중에서 CPU와 가장 가까워 가장 빠르다

CPU가 필요한 데이터를 메인메모리에서 찾기 전에 캐시메모리에서 먼저 찾는다.

캐시의 성능은 CPU가 찾으려는 내용이 많을수록 좋은 것

**메모리 계층 구조의 필요성** : CPU는 작은 메모리에 더 빨리 접근할 수 있다.

1. 디코딩 속도 : CPU가 빠르게 데이터에 접근하기 위해서는 데이터를 저장하는 메모리가 작아야 한다. 실제로 레지스터는 캐시보다 더 작은 용량을, 캐시는 메모리보다, 메모리는 하드 디스크보다 더 작은 용량을 가진다.
2. 자주 쓰는 데이터 : 자주 쓰이는 데이터는 계속 자주 쓰이고, 자주 쓰이지 않는 데이터는 계속 자주 쓰이지 않는다. 운영체제나 CPU는 자동으로 자주 쓰이는 데이터, 또는 자주 쓰일 것 같은 데이터를 메모리에서 캐시로 읽어온다. 자주 쓰이는 데이터는 전체 데이터 양에 비해 작은 양이기 때문에, 캐시는 메모리보다 더 작아도 된다.
3. 경제성 : 메모리 구조에서 상층에 속할 수록 더 비싸다. 비싼 하드웨어는 꼭 필요한 만큼의 크기만 사용하고, 싼 하드웨어를 넉넉한 크기만큼 사용하기 때문에 메모리 계층 구조가 피라미드 모양으로 나타나는 것이다.

# 운영체제 2. 프로세스

**프로그램과 프로세스** : 프로그램이 실행되려면 메인메모리에 들어가야 한다.

* 운영체제가 하드디스크에 있는 프로그램을 메인메모리로 옮긴다.
* 운영체제가 실행 프로그램에 대한 정보를 메모리에 저장하고 있으며, 프로그램을 관리한다.
* 프로세스 제어 블록(PCB) : 운영체제가 프로그램을 관리하는데 사용하는 정보

**프로세스** : 실행 중인 프로그램, 컴퓨터에서 연속적으로 실행되고 있는 컴퓨터 프로그램을 말한다.

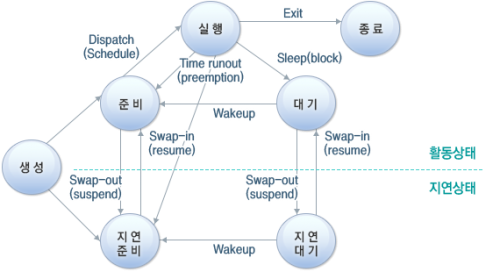
* 각 프로세스는 프로그램과 프로세스 제어 블록(PCB)로 구성

**프로세스의 상태**

* 생성상태(New) : 프로세스가 막 생성된 상태
* 준비상태(Ready) : 프로세스가 CPU에 실행되기 위해 대기하는 상태
* 실행상태(Run) : 프로세스에 포함된 명령어가 실행되고 있는 상태
* 대기상태(Wait) : 프로세스가 특정 자원이나 이벤트를 기다리는 상태
* 종료상태(Exit) : 프로세스가 실행을 완료한 상태

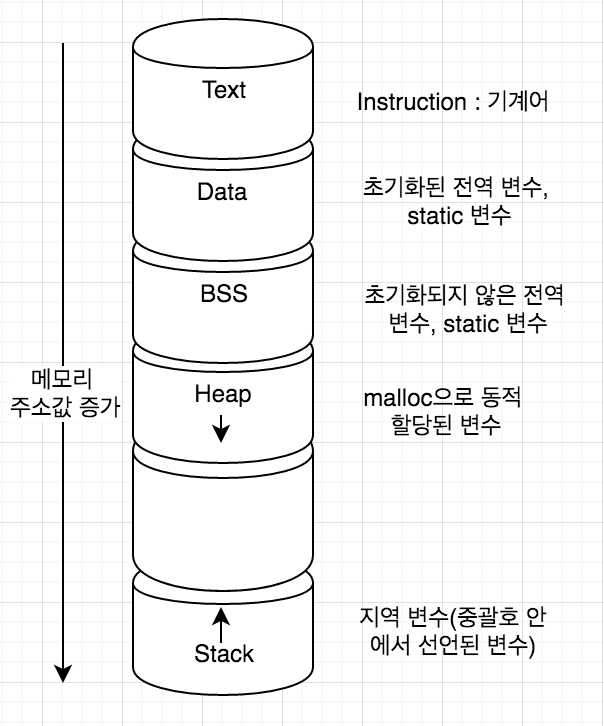
**프로세스의 동작**

* 준비->실행(Dispatch) : 우선순위가 높은 프로세스 선정하여 명령어 실행
* 실행->준비(Timer runout) : 인터럽팅 클록 발생시켜 제어권 빼앗음 (Preemption, 독점방지)
* 실행->대기(Block) : 프로세스가 입출력, 자원 등을 기다리기 위해 대기로 전환
* 대기->준비(Wake up) : 입출력이 완료되거나 자원이 할당되어 다시 실행
* Swap-out(Suspend) : 준비(대기) 상태에서 기억 장치를 반납하고 지연준비(지연대기) 상태로 전이
* Swap-in(Resume) : 지연준비(지연대기) 상태에서 기억 장치를 할당받아 준비(대기) 상태로 전이



* + 인터럽팅 클록 : 운영체제는 프로세스가 CPU를 자발적으로 반납하지 않고 독점하는 경우를 방지하기 위해 하드웨어적으로 인터럽팅 클록을 주기적으로 발생시켜 프로세스가 특정 시간 간격동안만 실행 할 수 있도록 한다. 인터럽팅 클록이 발생되면 강제로 실행중인 프로세스의 CPU제어권을 운영체제에게 빼앗기고 프로세스는 준비상태로 상태 전이 된다.

**프로세스의 메모리 공간**

스택영역과 힙영역 사이의 빈 공간 : 컴파일 타임에 지역변수를 얼마나 사용할지 미리 계산할 수 없기 때문에 런타임에 지역변수 선언 순서에 따라 스택영역 위쪽으로 주소값을 매기고 동적 할당될 때 힙영역 아래쪽으로 주소값을 매긴다.