<운영체제>

2. 운영체제 개념.

운영체제란? 컴퓨터 사용자 · 컴퓨터 하드웨어 사이 중계역할하고, 프로그램 실행관리하는 시스템 소프트웨어
→ 컴퓨터가 X, SW
컴퓨터의 자원을 효율적으로 관리 SW

운영체제 4가지 요소 1) resource 자원관리
2) 자원에 대한 exclusive 독점관리 요구
3) 관리자 supervisor
4) SW (커널과 OS, 디바이스드라이버로 구성)

운영체제 목적 1) 사용자의 컴퓨터 사용편리성
2) 컴퓨터의 자원관리 효율성.

운영 기능 (1) CPU · 프로세스 관리 (2) 메모리 관 (3) 파일시스템 등

운-체 VS 응용SW? 응용S → 사용자와 그밀. 사용자의 목적달성
운-체 → 하드웨어 등의 자원관리.
컴퓨터 콜 메커니 실행

2. 운영체제의 태동

① 1940	② 1945	③ 1950	④	⑤	⑥
고정 pro-	내장 pro	프로그램 코딩	로더의 필요성	일괄	최초의 운영

① 1940 운영체제 개념X · SW·HW 분리 X
→ ENIAC

② 1945 폰 노이만, 오늘날 · · , 펀치카드
CPU와 메모리 분리 / SW·HW 분리 /
프로그램 → 메모리, 프로그램실행 → CPU
→ HW 변화없이 프로그램만 메모리적재

③ 1950 IBM701 – 기계만 대여, SW제공X
개념시작 (첫카드) 로드프로그램을 적재. (뒤카드) 목적프로그램.
⇒ 시간 낭비를 줄이기위한 loader의 필요로 느낌.
⇒ 운. 체 발전

④ 1955 GM. 로더프로그램은 사용자 pro 분리.
→ 당시 로더를 모니터라고부름 GMOS 최초OS

⑤ 1956~ → 운영체제로 모습 갖춘 이유? 일괄방식으로 작동
최초 운체. 비슷한 작업들을 순서대로 메모리로딩& 한번에 하나 실행

3. 운영체제 발전 — 컴퓨터 시간이 귀함

1) 배치운영체제
∴ idle 시간을 줄여 효율↑
개발자 / 관리자 구분.
테이프 장치에 한 개의 프로그램 하나씩 적재·실행

2) 다중프로그래밍 운영체제
∴ CPU - I/O - CPU - I/O ··· I/O할동안 CPU가 대기
⇒ CPU의 idle 시간 줄이자.
여러 프로그램을 메모리에 적재! 프로그램 실행하다 I/O 발생시, 적재된 다른 프로그램을 실행하는 식임. (반응시간↓ queue)

3) 시분할 다중 프로그래밍 운영체제
∴ (배치+다중)문제 그거지 ① 비대화식처리 ② 응답 너무 느림
빠른프로그래밍 디버깅 & 대화식 시스템
CPU 활용 100% & 유휴시간 0

다중프로그래밍 이슈
메모리 이슈, 메모리 관리, 보호,
CPU 스케줄링, 컨텍스트 스위칭,
인터럽트, 동기화, 교착상태

2 컴터시스템

1. 컴터시스템 & HW

계층: 응용프로그램층 & 운·체층 & 컴퓨터 HW층

1) CPU (central processing unit)

프로그램코드(기계어명령) 해석 처리 중앙처리장치

전원공급시 시작, 메모리적재된 프로그램 실행

2) 메모리.

CPU에 의해 실행되는 코드 & 데이터 적재.

프로그램 메모리 적재 필수.

3) 캐쉬메모리

CPU 처리속도가 메모리보다 빠름 → CPU 대기시간 길어짐

⇒ 사이에 소량의 메모리설치.

on chip - 내부 off-chip - cpu외부.

캐쉬가 있으면, 대부분 캐쉬mem에서만 실행

4) bus.

하드웨어들이 데이터 수발신을 위해 디지털 신호가 지나가는 가닥의 전송 통로

통로로 묶어 부름. (주소 & 데이터 & 제어 등)

또는 목적에 따라 (1) 시스템버스(CPU나 메모리사이)(2) 입출력버스

상대적으로 빠름.

ATA : 디스크, USB 등 저장장치 연결 인터페이스

PCI : 메인보드와 주변장치(입출기) 연결

32비트 CPU?

1) CPU에 32개의 주소선 ⇒ 2^{32}개의 주소 ($0 \sim 2^{32}-1$번지)

1번지의 공간 = 1바이트(8비트), 2^{32}의 주소 = 2^{32}바이트, = 4GB.

2^{30} = 1GB X 32비트 CPU에 4GB 이상 메모리? X

2) CPU 입출력 32개의 데이터선

: 한번에 32개의 데이터선 사용 읽고쓰기 가능

명령: Instruction.

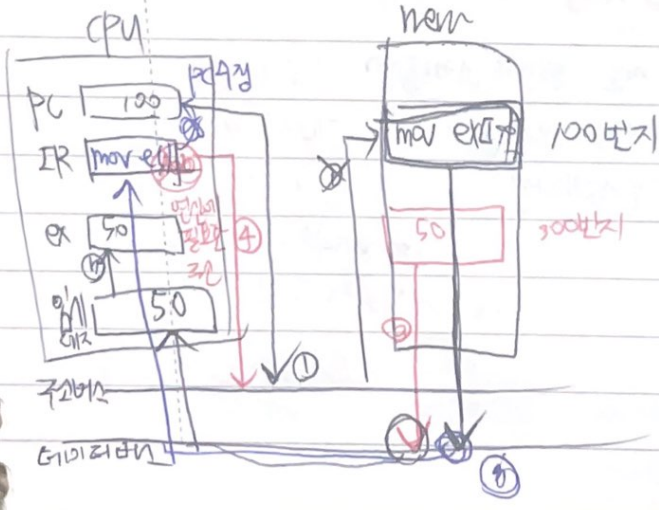
CPU 레지스터
- PC: 다음 실행할 명령어 메모리 주소 저장
- IR: 현재 실행하기 위해 메모리부터 읽어온 명령저장
- SP: 스택의 Top 주소 저장.

데이터 레지스터: 연산에 사용되거나 사용될 데이터 저장

상태 〃 : CPU의 상태정보 or 제어정보 (인터럽트등) 저장

명령어사이클: CPU 하나의 명령어를 실행하는 과정

<(내가 명령어를 실행하는 과정 11단계>



스택? : 운영체제에 의해 (프로그램마다) 메모리에 할당된 일부 영역
-> ~~별도~~ 하드웨어 메모리 X, 메모리 일부를 스택으로 사용
-> 스택공간할당, 프로그램마다

Context? 컨텍스트 : 프로그램이 실행 중인 상황 or 상황정보
(메모리, 레지스터값)

☆ 컨텍스트 스위칭? : 현재 실행중인 프로그램의 컨텍스트를 (CPU 레지스터값) 메모리에 저장
새로 실행시킬 // 저장된 // 를 다시 복귀

ㄴ 언제? CPU가 현재 프로그램실행 중지하고, 다른 프로그램 실행시

멀티코어 CPU : CPU 내부의 2개의 프로세서…

2. 컴퓨터시스템과 운·체

왜 계층구조? "계층간 독립성 확보"
사용자는 운·체몰라도 OK
하드웨어몰라도 개발 OK
하/ 바뀌어도 응용은 재작성 X

<운영체제 필요성 5가지>

(2명) 동시에 pr-m을 실행시 스케줄링? 프로세스관리
메모리 사용? 종료된 프로그램의 메모리 더슨? 메모리 //
(//) 동시 프린팅 -> 충돌안 생기려면? 장치 //
파일기록시 어디에? 파일시스템 //
키보드? 입출력관리

=> 자원충돌 최소화 · 성능최적화 · 사용자 시스템 사용 효율화

응용프로그램에 운영체제 역할? | 응용프로그램이 직접 하드웨어를 다루지 못하게 차단.
→ 〃 사이의 하드웨어 사용 충돌을 막기위해.

응용프가 하드웨어 사용하려고 할 때…

"시스템호출" | 반드시 운영체제에 요청 → 운영체제가 대신 HW조작.
→ 소프트웨어인터럽트 / 응용프의 인터페이스
→ 인터럽트? 컴퓨터 하드웨어 간의 〃

사용자에 운·체 역할. | HW지식없어도 컴퓨터 사용 용이. 인터페이스 제공

하드웨어 운·체 | 하드웨어를 제어하는건 운영체제 몫. (입출력 모두 운체가)

운영체제 구성. | "커널 + 툴 + 디바이스 드라이버"
커널 - 핵심. 코드와 데이터로 이루어짐 (부팅후 메모리상주)
커널기능 이용시 응용프로그램은 반드시 시스템호출 사용.

툴 - 관리. 쉘. 제어판 ..

디바이스 드라이버 - 장치직접제어 & 입출력 SW

↗ 사용자공간코드에서 커널서비스를 요청하는 방법

〈커널이 제공하는 2개의 인터페이스: 시스템호출 & 인터럽트〉
→ 응용프로그램과 HW의 중계역할.

(1) 시스템호출 system call (커널 - 응용프로그램 인터페이스)
- 유일하게 응용프로그램에서 커널 기능을 사용할 수 있는 방법
- 시스템호출 라이브러리를 통해 시스템호출함수 제공. (운·체 패키지 포함)
ex 파일읽기, 메모리할당…

(2) 인터럽트 (커널 - HW 인터페이스)
- 장치들이 입출력완료, 타이머완료 등을 CPU에게 알림.
- 인터럽트 발생 → CPU: 응중단 → 인터럽트서비스루틴 ㄱㄱ

3. 커널 & System Call - 오늘 주제는 다중 프로그래밍

→ 다수응용프로그램 하나의 컴퓨터실행.

문제) 응용프로그램이 직접 컴퓨터 자원접근 → 충돌·위험

커널) 이 자원접근 불허.

→ 자원접근은 커널에만 한한부여. (How?)

(1) 사용자 공간 & 커널공간.

컴퓨터 메모리 공간을 나눔.

- 사용자공간: 모든 응용프로그램이 나누어 씀. 작지
- 커널 〃 : 커널만 사용. 디바이스드라이브 포함.

왜 나눔? (커널코드나 데이터를) 악의적인 응용프로그램으로부터 보호

사용자공간크기: 한 응용프로그램의 최대크기 (프로그램코드 + 데이터(전역변수) + 동적할당 + 스택)

〃 주소공간범위: 응용프로그램은 사용자 공간의 주소범위를 넘어설 수 없음

주소공간 = 사용자 공간 + 커널공간

↳ 가상주소공간 (물리 메모리주소와 무관)

↳ 사용자 or 응용프로그램 관점에서 보는 주소범위.

↳ 사용자가 전체 메모리를 사용한다착각.

모든 응용프로그램은 커널공간을 공유함.

(2) 사용자모드 & 커널모드 CPU는 둘 중 한모드실행

- 사용자모드 = CPU의 모드비트 1
  - CPU는 사용자프로그램 코드나 데이터 억세스중
  - 〃 커널공간불허 응용프로그램으로부터 커널방어모드
  - HW. 다른응용프로그램에 할당된 메모리접근 불허
  - 특권명령 실행불허 (입출력장치등 하드웨어 조작명령등 시스템인터럽터 특별명령)

(비고를 p.46)

- 커널모드 = CPU 모드비트 0
  - CPU가 커널공간에서 실행될 동안, 혹은 사용자코드를 실행하는 동안
  - 특권명령 사용 OK

< 사용자모드 → 커널모드 : 시스템콜 & 인터럽트 발생 >

- 시스템콜: 시스템콜을 실행하는 기계명령에 의해 수행
  기계명령이 CPU의 모드비트를 커널모드(0)으로 전환

- 인터럽트: CPU가 인터럽트를 수신하면 커널모드로 전환된다.
  → 인터럽트 서비스루틴이 커널공간에 있으므로.
  CPU는 인터럽트 서비스루틴을 실행 후, 끝나면 사용자모드로 바꿔준다

특권명령: 커널모드에서만 실행할 목적으로 설계된 CPU명령
I/O, Halt (작업중지, CPU를 유휴상태로 만듦), 메모리보호
인터럽트 플래그를 키고끄는, 타이머, 컨텍스트스위칭; 기타등

p.2의 문제) 특권명령 같이 읽어보기

{ 커널은 부팅시에 커널공간에 적재된 함수들과 데이터변수
  커널코드 = 함수의 집합 } ☆

< 응용프로그램이 활용하는 라이브러리는 2가지 유형 >

① 표준라이브러리: 사용법동일
② 시스템콜 라이브러리: 운영체제 비종속
  시스템콜함수를 커널API라고 함

라이브러리! 응용프로그램에서 활용하도록 미리 함수들을 작성하고 컴파일하여
바이너리 형태로 제공 확인

함수호출 : 사용자공간에서 ~~정의된함수~~, 사용자모드로 실행
// 에 작재함수가 다른함수 ⓐ 라이브러리함수 로
- 호출된 함수가 끝나면, 호출한 곳으로 복귀.

시스템콜 : 응용프로그램이 운영체제의 기능을 사용할 경우.
사용자모드에서 커널모드로 변경된 후 커널함수 실행
- 시스템 호출을 일으키는 특별한 기계명령실행

사용자가 (작성된 코드와 라이브러리 함수들) → 링크되어 ⇒ 하나의실행파일

함수호출로 라이브러리 활용 / 시스템콜로 커널코드실행
→ 사용자공간에 작재된 함수호출함으로 → 커널함수에서 리턴될때, 사용자모드전환 및 프로그램으로 복귀

4. 운영체제 & 인터럽트

인터럽트 : 입출력장치들이 비동기 사건을 CPU에게 알리는 신호
① 실행중 ② 인터럽트발생 ③ 인터럽트서비스중.
④ CPU INTR 인터럽트신호발생 ⑤ 인터럽트벡터번호 N
⑥ 처리루틴 ISR의 주소알아냄 ⑦ 인터럽트서비스루틴실행

입출력 신호일때, 장치로부터 입출력완료 통보를 받는 방법.
→ 인터럽트.
없다면, CPU는 입출력완료를 계속감시하는 폴링실행 = 비효율적.
// 다중 프로그래밍 운영체제 구현 불가능.
⇒ 입출력 장치와 CPU가 동시에 각자 작업실행
⇒ 컴퓨터 시스템의 효율적작동. (CPU활용, 시스템처리율 ↑)