3차시 수업

드디어 시작!

- 우선, 어셈블리와 레지스터에 대해 간단히 알아봅시다.
- 어셈블리는 나오는 것만 간단하게 배울 예정
- 레지스터는 우선 16비트만 배울 예정
- 간단한 부트로더를 구현해봅시다.

Main registers

	AH	AL	**AX** (primary accumulator)
	BH	BL	**BX** (base, accumulator)
	CH	CL	**CX** (counter, accumulator)
	DH	DL	**DX** (accumulator, other functions)

Index registers

0 0 0 0	SI	**S**ource **I**ndex
0 0 0 0	DI	**D**estination **I**ndex
0 0 0 0	BP	**B**ase **P**ointer
0 0 0 0	SP	**S**tack **P**ointer

Program counter

0 0 0 0	IP	**I**nstruction **P**ointer

Segment registers

CS	0 0 0 0	**C**ode **S**egment
DS	0 0 0 0	**D**ata **S**egment
ES	0 0 0 0	**E**xtra**S**egment
SS	0 0 0 0	**S**tack **S**egment

Status register

	-	-	-	-	O	D	I	T	S	Z	-	A	-	P	-	C	Flags

일반 레지스터

- AX (AH + AL) : 입출력과 산술 연산에서 주로 사용
- BX (BH + BL) : 인덱스로 사용가능한 유일한 레지스터, 색인용
- CX (CH + CL) : 반복 횟수, 비트 이동 횟수 등의 횟수 제어용
- DX (DH + DL) : 특정 입출력 시 필수 사용됨, AX와 함께 산술용

세그먼트 레지스터

- CS : 프로그램에서 코드가 쓰이는 곳의 시작 주소를 가리킴
- DS : 프로그램에서 데이터가 쓰이는 곳의 시작 주소를 가리킴
- SS : 메모리 상의 스택 구현용
- ES, FS, GS : 여분의 세그먼트 레지스터

주소 레지스터

- BP : 스택의 시작 주소를 가리킴
- SP : 스택의 현 위치를 가리킴
- IP : 다음에 실행될 명령어의 오프셋 주소를 가리킴
- SI : 출발지 주소를 가리킴
- DI : 도착지 주소를 가리킴

플래그 레지스터

- OF(Overflow) : 연산 결과가 범위를 넘어 섰는가? 1: 0;
- DF(Direction) : 문자열 조작에서 주소 레지스터 값이 감소하는가? 1: 0;
- IF(Interrupt) : 인터럽트 요구를 받아들일 수 있는가? 1: 0;
- TF(Trap) : 한 명령어가 실행될 때 마다 인터럽트 되는가? 1: 0;
- SF(Sign) : 연산 결과 음수인가? 1: 0;
- ZF(Zero) : 연산 결과 0인가? 1: 0;
- AF(Auxiliary) : 피연산자의 비트3에서 비트4로 자리올림 또는 비트4에서 비트3으로 자리내림이 일어났는가? 1; 0:
- PF(Parity) : 연산 결과 1비트가 짝수 개인가? 1: 0;
- CF(Carry) : 산술 결과 가장 왼편의 비트에서 빌림이나 올림이 일어났는가? 1: 0;

어셈블리

- 오늘 나오는 명령어
- db, dw, dd, org, jmp, mov, resb, add, cmp, je, int, hlt, times
- 오늘 나오는 심볼과 데이터 형
- label, $, $$, byte, word, dword

어셈블리 - 데이터 크기

- BYTE = 8비트 데이터
- WORD = 16비트 데이터
- DWORD = 32비트 데이터

- 참고로, Intel 80x86 플랫폼에선 리틀 엔디안 방식 사용

어셈블리 - 명령어 (1)

- db, dw, dd : 각각 byte, word, dword 단위로 파일의 내용을 씀
- org {var} : {var} 번지로 프로그램 시작 주소를 변경 시킴
- jmp {var} : goto {var}; {var}로 이동하시오
- resb {var} : {var}개의 바이트 만큼 띄어 놓으시오
- add {var1}, {var2} : {var1}에 {var2}를 더하시오
- cmp {var1}, {var2} : var1과 var2를 비교하시오 (오직 비교만)

어셈블리 – 명령어 (2)

- je {var} : 비교의 결과가 ‘같다’ 이면 {var}로 점프하시오
- int {var} : {var}번째의 소프트웨어 인터럽트 명령을 내리시오.
  나중에 자세히 알려줌.
- hit : cpu를 정지시키시오 (대기 상태로 만드시오)
- times {var} : {var}번 만큼 반복하시오
- mov {var1} {var2} : {var1} = {var2}; {var1}을 {var2}로 대입하시오.
- mov 연산자는 []연산자를 통하여 주소도 대입 가능

어셈블리 – 다양한 기호

- $: 현재의 위치
- $$: 현재 프로그램의 시작 주소
- label : 명령어들의 위치를 나타내는 기호

다음 시간 예고

- 커널 제작
- 부트로더에서 커널 로드
- C언어로 그래픽 요소 구현