C basic language

임베디드스쿨 2기

Lv1과정

2021. 04. 02

김효창

CPU 구조

레지스터 : CPU 내부에 존재하며 직접적으로 ALU에 연결되어 입·출력 값을 저장하는 역할

(RAM 은 CPU에서 어드레스 라인으로 주소를 지정하고 해당 주소의 데이터를 읽고 쓰는 방식)

범용 레지스터 : 산술 · 논리 연산, 주소에 사용되는 피연산자 , 메모리 포인터가 저장되는 곳

세그먼트 레지스터 : code , data , stack의 지시하는 주소를 저장

플래그 레지스터 : 프로그램의 현재 상태나 조건 등을 검사하는 데 사용

instruction 포인터 : 다음 수행해야 하는 명령이 있는 메모리 주소를 저장

1. 범용 레지스터

AX : 산술 연산 , system call, 함수 리턴 값, 피연산자의 크기에 따라 AX 또는 AL 레지스터에 저장

BX : 데이터 주소를 지시하는 포인터

CX : 시프트/벡터 연산과 루프(반복문), counter 에서 사용

DX : I/O 포인터 , 부호 확장 명령에 사용,

큰 값을 포함하는 작업을 곱하고 분할하기 위해 DX와 함께 AX 레지스터와 함께 사용

SP : 메모리 stack 의 마지막 주소를 지시

BP : 메모리 stack 의 첫 시작 주소를 지시

DI , SI : 문자/문자열 이동/비교 에 사용

CPU 구조

2. 플래그 레지스터

1) Status flags

0x0000 00002 : 시스템 초기화

Carry flag : 자리 올림 or 자리 내림이 발생하면 SET (1)

Parity flag : 연산 결과 1 Bit 들의 개수를 나타낸다 . 짝수 = even parity , 홀수 = odd parity

Adjust flag : 자리올림 or 자리 내림이 3 Bit 이상 발생할 경우 SET (1)

Zero flag : 산술 또는 비교(if문) 연산의 결과를 나타냅니다.

결과 0 이면 SET (1), 결과 0 아니면 RESET (0)

Sign flag : signed 변수인 경우 양수 = 0 , 음수 = 1

Overflow flag : 데이터 범위에서 초과하거나 미만인 경우 SET (1)

Interrupt enable flag : 1 (인터럽트 허용) , 0 (인터럽트 금지)

Trap flag : 디버깅 할 때 single step 허용하려면 SET (1)

사용하는 DEBUG 프로그램은 트랩 플래그를 설정하므로 한 번에 하나의 명령을 실행할 수 있다

Direction Flag : 문자열 데이터를 이동하거나 비교할 때 왼쪽 또는 오른쪽 방향을 결정합니다.

DF 값이 0이면 문자열 연산은 왼쪽에서 오른쪽 방향으로 이동

값이 1로 설정되면 문자열 연산은 오른쪽에서 왼쪽 방향으로 이동

CPU 구조

3. 세그먼트 레지스터

Code Segment (CS) : 명령어 저장 공간

Data Segment (DS) : 변수 저장 공간

Stack Segment (SS) : 함수 저장 공간

Extra segment (ES) : 프로그래머가 정해서 쓰는 공간

FS, GS : 80386(32bit) CPU 때 추가된 정해지지 않는 영역

3. CPU의 할당 받는 개념

CPU 에는 IP 레지스터가 존재

IP 레지스터가 코드 영역에 있는 것을 지시하면 CPU 는 할당 받은 상태가 되고 실행 상태가 된다

동시에 프로세서가 추가로 실행되면 이것은 할당 받지 않은 상태여서 대기 상태이다

IP : 명령어가 있는 코드 영역을 지시,

현재 프로그램에서 몇 번째 실행 코드를 실행 했는가의 정보를 저장하는 레지스터

CPU는 Instruction Point 가 지시하는 위치에서 명령어를 가지고 온다

플래그 레지스터 : 비교 및 쉬프트 연산 후에 발생되는 결과를 저장하는 레지스터

비교해서 참인 경우 특정 비트 값을 1로 설정

CPU 구조

그림 1

SN54LS181, SN54S181 . . . J OR W PACKAGE
SN74LS181, SN74S181 . . . DW OR N PACKAGE
(TOP VIEW)

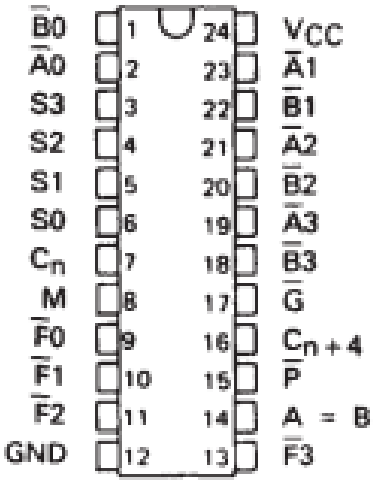

그림 2

TABLE 1

SELECTION				ACTIVE-LOW DATA		
				M = H	M = L; ARITHMETIC OPERATIONS	
S3	S2	S1	S0	LOGIC FUNCTIONS	Cn = L (no carry)	Cn = H (with carry)
L	L	L	L	F = $\overline{A}$	F = A MINUS 1	F = A
L	L	L	H	F = $\overline{AB}$	F = AB MINUS 1	F = AB
L	L	H	L	F = $\overline{A}$ + B	F = A$\overline{B}$ MINUS 1	F = A$\overline{B}$
L	L	H	H	F = 1	F = MINUS 1 (2's COMP)	F = ZERO
L	H	L	L	F = $\overline{A + B}$	F = A PLUS (A + $\overline{B}$)	F = A PLUS (A + $\overline{B}$) PLUS 1
L	H	L	H	F = $\overline{B}$	F = AB PLUS (A + $\overline{B}$)	F = AB PLUS (A + $\overline{B}$) PLUS 1
L	H	H	L	F = $\overline{A \oplus B}$	F = A MINUS B MINUS 1	F = A MINUS B
L	H	H	H	F = A + $\overline{B}$	F = A + $\overline{B}$	F = (A + $\overline{B}$) PLUS 1
H	L	L	L	F = $\overline{AB}$	F = A PLUS (A + B)	F = A PLUS (A + B) PLUS 1
H	L	L	H	F = A ⊕ B	F = A PLUS B	F = A PLUS B PLUS 1
H	L	H	L	F = B	F = A$\overline{B}$ PLUS (A + B)	F = A$\overline{B}$ PLUS (A + B) PLUS 1
H	L	H	H	F = A + B	F = (A + B)	F = (A + B) PLUS 1
H	H	L	L	F = 0	F = A PLUS A‡	F = A PLUS A PLUS 1
H	H	L	H	F = A$\overline{B}$	F = AB PLUS A	F = AB PLUS A PLUS 1
H	H	H	L	F = AB	F = A$\overline{B}$ PLUS A	F = A$\overline{B}$ PLUS A PLUS 1
H	H	H	H	F = A	F = A	F = A PLUS 1

‡Each bit is shifted to the next more significant position.

그림 3

M = H LOGIC FUNCTIONS
F = $\overline{A}$
F = $\overline{A + B}$
F = $\overline{A}$B
F = 0
F = $\overline{AB}$
F = $\overline{B}$
F = A ⊕ B
F = A$\overline{B}$
F = $\overline{A}$ + B
F = $\overline{A \oplus B}$
F = B
F = AB
F = 1
F = A + $\overline{B}$
F = A + B
F = A

CPU 구조

그림 1

1 ~ 2 , 18 ~ 23 pin: A 입력 , B 입력 (각 4 Bit)

3 ~ 6 pin : select 핀 (어떤 연산을 실행할 것인지 결정)

7 pin : 아래에서의 자리 올림 (carry)

8 pin : mode pin (산술 , 논리연산의 종류를 결정)

9 ~ 11, 13 pin : 출력 4 Bit

그림 2

M 은 모드 핀 (M = H 논리 연산 , M = L 산술 연산)

산술 연산은 자리 올림 carry 가 있는지 없는지 구분되어 있음 (Cn = H 캐리 없다, Cn = L 캐리 있다)

S 는 select (4개의 조합 16가지에 의해 여러 종류 연산 가능)

CPU 구조

M = H LOGIC FUNCTIONS	
1	$F = \overline{A}$
3	$F = \overline{A + B}$
4	$F = \overline{A}B$
5	$F = 0$
6	$F = \overline{AB}$
9	$F = \overline{B}$
10	$F = A \oplus B$
11	$F = A\overline{B}$
12	$F = \overline{A} + B$
14	$F = \overline{A \oplus B}$
15	$F = B$
	$F = AB$
	$F = 1$
	$F = A + \overline{B}$
	$F = A + B$
	$F = A$

Cn = L (no carry)	Cn = H (with carry)
F = A MINUS 1	F = A
F = AB MINUS 1	F = AB
F = $A\overline{B}$ MINUS 1	F = $A\overline{B}$
F = MINUS 1 (2's COMP)	F = ZERO
F = A PLUS (A + $\overline{B}$)	F = A PLUS (A + $\overline{B}$) PLUS 1
F = AB PLUS (A + $\overline{B}$)	F = AB PLUS (A + $\overline{B}$) PLUS 1
F = A MINUS B MINUS 1	F = A MINUS B
F = A + B	F = (A + $\overline{B}$) PLUS 1
F = A PLUS (A + B)	F = A PLUS (A + B) PLUS 1
F = A PLUS B	F = A PLUS B PLUS 1

1번 : 연산 결과 F는 A와 B 각 비트끼리 OR의 NOT을 연산

3번 : 연산 결과 F는 입력에 관계없이 0 이 된다

4번 : F는 A와 B 각 비트끼리 AND의 NOT을 연산

5번 : 연산 결과 F는 B의 NOT , B의 모든 비트의 1과 0을 반전한 것

6번 : F는 A와 B 각 비트끼리 EXOR 연산한 것

9번 : F는 A와 B 각 비트끼리 EXOR의 NOT을 연산한 것

10번 : 연산 결과 F는 B이다

12번 : 연산 결과 F는 입력에 관계없이 모든 비트에 1을 넣는 것

15번 : 연산 결과 F는 A이다

빨간색

캐리가 없는 경우 : (A – B) - 1

캐리가 있는 경우 : F = A – B

파란색

캐리가 없는 경우 : F = A + B

캐리가 있는 경우 : F = (A + B) + 1

매개 변수 : 함수 호출 시 전달되는 인자를 저장할 변수

함수 호출 시 전달되는 인자의 수는 여러 개가 될 수 있지만 반환할 수 있는 값의 개수는 최대 1개

1. 지역 변수

중괄호에 내 에 선언되는 변수는 모두 지역 변수

지역 변수는 선언된 지역 내에서만 유효하다

지역 변수는 함수가 호출될 때 메모리에 기록되고 , 함수가 종료되면 메모리에서 지워진다

지역 변수는 외부에 선언된 동일한 이름의 변수를 가릴 수 있다. (전역변수보다 "지역 변수" 우선)

모든 매개 변수는 지역 변수이다

2. 전역 변수

어디에서나 접근이 가능한 변수

초기값 정하지 않아도 0으로 초기화

C 프로그램 Build → 컴파일 (object file : source file을 기계어로 변환), 링킹 (exe 파일 생성)

→ 실행 파일 확보 (코드, 데이터 영역) → 코드와 데이터를 실행하면 커널에 loader 함수가 동작

(실행 파일을 메모리에 상주) → 스택 영역 생성 → 힙 영역 생성 (런타임, 동적 메모리)

프로그래밍 시 코드 영역, 데이터 영역, 스택 영역, 힙 영역의 메모리 공간 확보하고
CPU를 할당 받아서 코드를 수행함으로써, 상기 영역에 데이터를 쓰거나 읽을 수 있다.