강의	정보처리 필기	강사	조대호
차시 명	[CA-07강] 주소지정방식과, 명령실행과 제어	차시	7차시

학습내용

☞ 주소지정방식과 연산
명령실행과 제어

학습목표

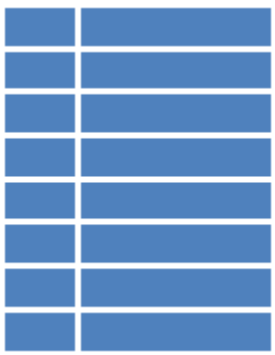
☞ 주소지정방식과 연산에 대해 이해 할 수 있다
명령실행과 제어를 이해하고 해결 할 수 있다

학습내용

1. 주소지정방식(Addressing Mode)

암시적(묵시적) 주소지정 방식(Implied Addressing Mode)
주소를 지정하는 필드가 없는 0번지 명령어에서 Stack의 Top 포인터가 가리키는 Operand를 암시하여 이용함.

즉시적 주소지정 방식(Immediate Addressing Mode)
Operand 부분에 데이터를 기억하는 방식, 기억장치를 접근하지 않음.



직접 주소지정 방식(Indirect Addressing Model)
명령어 주소 부분에 유효 주소 데이터가 있음, 데이터 값 범위가 제한적.

간접 주소지정 방식(Indirect Addressing Mode)
명령문 내의 번지는 실제 데이터의 위치를 찾을 수 있는 번지가 들어 있는 장소를 표시

계산에 의한 주소지정 방식
Operand 부와 CPU의 특정 레지스터의 값이 더해져서 유효주소를 계산
프로그램이 수행되는 동안 사용될 데이터의 위치를 지정하는 방법
- 설계 시 고려사항 : 표현의 효율성, 사용의 편리성, 주소공간과 기억공간의 독립성

종 류	설 명
상대 주소 지정 방식 (Relative Addressing Mode)	• 유효주소: 명령어의 주소 부분 + Program Counter • 명령어 자신의 기억장소를 기준으로 하여 데이터의 위치를 지정하는 방식
베이스 레지스터 주소 지정 방식 (Base Register Addressing Mode)	• 유효주소: 명령어의 주소 부분 + Base Register • 프로그램을 재배치(Relocation)할 때 이용 • 다중 프로그래밍 기법에 많이 사용 ※베이스 레지스터 : 명령이 시작되는 최초의 번지를 기억하고 있는 레지스터
인덱스 레지스터 주소 지정 방식 (Indexed Addressing Mode)	• 유효주소: 명령어의 주소 부분 + Index Register

리엔트란시(Re-entrancy)
한 사람 이상의 사용자들이 동일한 명령어를 동시에 실행할 수 있도록 제공
인덱스 레지스터와 간접번지 방법으로 이용

2. 연산(Operation)

논리연산
연산의 대상 및 결과가 '0'또는 '1'중 하나의 값을 취하는 연산(비수치적인 연산)
예> MOVE, NOT, AND, OR, 논리 SHIFT, ROTATE, COMPLEMENT, EXCLUSIVE OR 등

산술연산
연산의 대상을 수치 데이터로 간주하고 행하는 연산(수치적 연산)
예> ADD, SUBTRACT, MULTIPLY, DIVIDE, 산술 SHIFT 등

주요 연산 및 기능
AND(Masking Operation) : 마스크를 이용하여 불필요한 부분을 제거
OR(Selective-Set) : 두 개의 데이터를 섞거나 일부에 삽입
XOR(Compare) : 자료의 특정 비트를 반전시키고자 하는 경우에 사용
NOT(Complement) : 각 비트의 값을 반전시키는 연산
논리SHIFT : 왼쪽 또는 오른쪽으로 1bit씩 자리를 이동시키는 연산
ROTATE : 논리 Shift에서 밀려 나가는 비트의 값을 반대편 값으로 입력하는 연산

3. 산술 Shift

부호비트를 제외한 나머지 비트만 Shift, 왼쪽으로 n Bit Shift
왼쪽으로 n Bit Shift : 원래 자료에 2의 n승을 곱한것과 같음
오른쪽으로 n Bit Shift : 원래 자료를 2의 n승으로 나눈 값과 같음

Shift	수치 표현법	Padding Bit	
		음수	양수
Shift Left	부호화 절대치	Padding Bit : 0	모든 Padding Bit는 0
	1의 보수법	Padding Bit : 1	
	2의 보수법	Padding Bit : 0	
Shift Right	부호화 절대치	Padding Bit : 0	
	1의 보수법	Padding Bit : 1	
	2의 보수법	Padding Bit : 1	

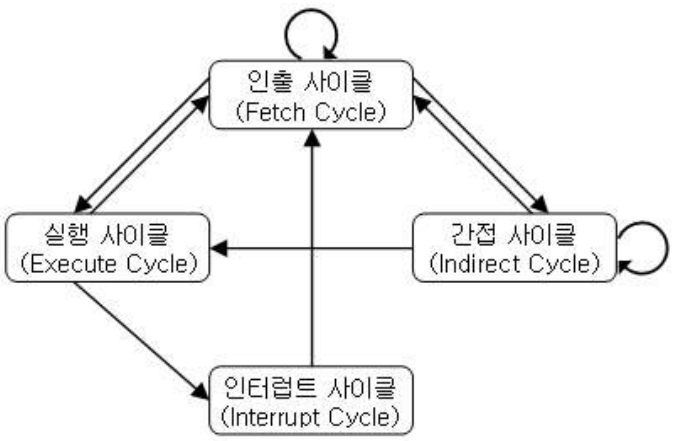
※ Padding Bit : Shift에서 자리를 이동한 후 생기는 왼쪽 혹은 오른쪽 끝의 빈자리에 채워지는 비트

4. 명령실행과 제어

마이크로 오퍼레이션(Micro Operation)
명령을 수행하기 위해 CPU 내의 레지스터와 플래그의 상태 변환을 일으키는 작업
제어신호 : 마이크로 오퍼레이션을 순차적으로 일어나게 하는데 필요한 신호.
마이크로 오퍼레이션은 Clock 펄스에 기준을 두고 실행
기억장치로부터 명령어를 인출하여 해독하고, 해독된 명령어를 실행하기 위해 제어 신호를 발생시키는 각 단계의 세부 동작을 말함
* 동기 디지털 시스템에 내장되어 있는 모든 레지스터의 타이밍은 마스트 클록 발생기에 의하여 제어됨

마이크로 사이클 타임(Micro Cycle Time)
마이크로 오퍼레이션 수행에 필요한 시간을 마이크로 사이클 타임이라 함
마이크로 사이클 타임 부여 방식
- 동기 고정식(Synchronous Fixed) : 수행시간이 가장 긴 것을 정의한 방식 처리속도의 길이가 비슷할 때 사용
- 동기 가변식(Synchronous Variable) : 수행시간을 다르게 하는 것(CPU를 효율적 사용 가능)
- 비동기식(Asynchronous) : 서로 다른 마이크로 사이클 타임을 정의, 제어기가 복잡해 실제로는 거의 사용되지 않음.

5. 메이저 스테이트(메이저 상태, Major State)
: 정의 : CPU가 무엇을 하고 있는가를 나타내는 것으로서 기억 장치의 사이클을 단위로 하여 해당 사이클 동안에 무엇을 위해 기억 장치를 접근하는가를 나타내 주는 것



간접 단계(Indirect Cycle)
인스트럭션의 수행 시 유효 주소를 구하기 위한 메이저 상태
간접 단계(Indirect cycle)동안에 기억장치로 부터 오퍼랜드(데이터)의 번지(Address)를 인출

실행 단계(Execute Cycle)
실제로 명령을 이행하는 단계

인출 단계(Fetch Cycle)
주기억 장치의 지정 장소로 부터 명령을 읽어서 중앙처리장치에 가지고 오는 단계
인터럽트를 처리한 후 다음으로 전환해야 될 메이저 스테이트

T1 : MAR ← PC	• PC에 있는 번지를 MAR에 전송
T2 : MBR ← M[MAR], PC ← PC+1	• 메모리에서 MAR이 지정하는 위치의 값을 MBR에 전송 • 다음에 실행할 명령의 위치를 지정하기 위해 PC의 값을 1증가시킴
T3 : OPR ← MBR[OP], I ← MBR[I]	• 명령어의 Op-Code 부분을 명령 레지스터에 전송 • 명령어의 모드 비트를 플립플롭 I에 전송
T4 : F ← 1 또는 R ← 1	• I가 0이면 F 플립플롭에 1을 전송하여 실행 사이클로 변하고, I가 1이면 R 플립플롭에 1을 전송하여 간접 사이클로 변함

인출 단계(Fetch Cycle)에서 일어나는 마이크로 오퍼레이션

인터럽트 단계(Interrupt Cycle)
하드웨어로 실현되는 서브루틴의 호출이라고 볼 수 있음
Interrupt Cycle에서 일어나는 마이크로 오퍼레이션

```
1. MBR(AD) ← PC,   PC ← 0
2. MAR ← PC,   PC ← PC+1
3. M ← MBR,   IEN ← 0
4. F ← 0,   R ← 0
```

인터럽트 단계에서 일어나는 마이크로 오퍼레이션

6. 주요 명령의 마이크로 오퍼레이션

1) AND
① AC(누산기) 내용과 메모리 내용을 AND(논리곱) 연산하여 결과를 AC에 저장하는 연산 명령
② 마이크로 오퍼레이션

```
MAR ← MBR
MBR ← M(MAR)
AC ← AC AND MBR
```

2) ADD

① AC와 메모리의 내용을 더하여 결과를 AC에 저장하는 연산 명령

② 마이크로 오퍼레이션

```
MAR ← MBR(ADDR)
MBR ← M(MAR)
AC ← AC + MBR
```

3) LDA(Load to AC)

① 메모리의 내용을 AC로 가져오는 명령

② 마이크로 오퍼레이션

```
MAR ← MBR(AD)
MBR ← M(MAR),  AC ← 0
AC ← AC + MBR
```

4) STA(Store AC)

① AC의 내용을 메모리에 저장하는 명령

② 마이크로 오퍼레이션

```
MAR ← MBR(AD)
MBR ← AC
M ← MBR
```

5) BUN(Branch UNconditionally)

① PC에 특정한 주소를 전송하여 실행명령의 위치를 변경하는 무조건 분기 명령

② 마이크로 오퍼레이션

```
PC ← MBR(AD)
```

6) BSA(Branch and Save Return Address)

복귀주소를 저장하고 부 프로그램을 호출하는 명령

7) ISZ(Increment and Skip if Zero)

① 메모리의 값을 읽고 그 값을 1 증가시킨 후 음수에서 시작한 그 값이 0이면 현재 명령을 건너 띄고 다음 명령으로 이동

② 마이크로 오퍼레이션

```
MAR ← MBR(AD)
MBR ← M
MBR ← MBR + 1
M ← MBR, IF(MBR=0) THEN (PC←PC+1)
```

7. 제어장치

제어 데이터의 종류

각 메이저 스테이트 사이의 변천을 제어하는 제어 데이터

중앙처리장치의 제어점을 제어하는데 필요한 제어 데이터

인스트럭션의 수행순서를 결정하는데 필요한 제어 데이터

제어 데이터가 될 수 있는 것
연산자의 종류, 인스트럭션의 주소지정방식, 연산 결과에 대한 상태플래그 내용

제어장치(제어기)의 구현
고정 배선 방식 : 속도가 빠르지만, 마이크로프로그램 방식에 비해 가격이 비쌈
마이크로 프로그램 방식 : 고정 배선 방식에 비해 속도가 느림

구 분	고정배선 제어장치	마이크로 프로그램 방식
반응 속도	고속	저속
회로 복잡도	복잡	간단
경제성	비경제적	경제적
융통성	없음	있음
구성	하드웨어	소프트웨어
제어 메모리	불필요	필요

8. 마이크로 프로그램(Microprogram)
명령을 수행할 수 있도록 된 일련의 제어 워드가 특수한 기억 장치 속에 저장된 것으로 각종 제어신호를 발생시킴
마이크로 명령으로 형성되어 있음
전자계산기의 제작 단계에서 컨트롤 스토레이지 속에 저장됨
제어기를 구성하는 방법으로 마이크로프로그램이 이용될 수 있음
마이크로프로그램이 저장되는 제어 메모리는 ROM에 주로 사용되고 사용자가 변경시킬 수 없음
별도의 번역과 RAM으로 접근이 필요하지 않기 때문에 일반적인 소프트웨어와 구별하여 펌웨어(Firmware)라고도 함

요점정리

1. 주소지정방식과 연산에 대해 정리합니다.
2. 명령실행과 제어에 대해 정리합니다.

다음차시예고

수고하셨습니다. 다음 8주차에서는 "[CA-8강] 주기억장치와 보조기억장치"에 대해서 학습하도록 하겠습니다.