6강

- PSR Register 속성
  - condition flag : ALU 연산 직후 그 결과 값을 나타냄
  - control bits : CPU의 상태를 나타냄

    I/F, T, Mode
- 메모리 구조
  - Endian

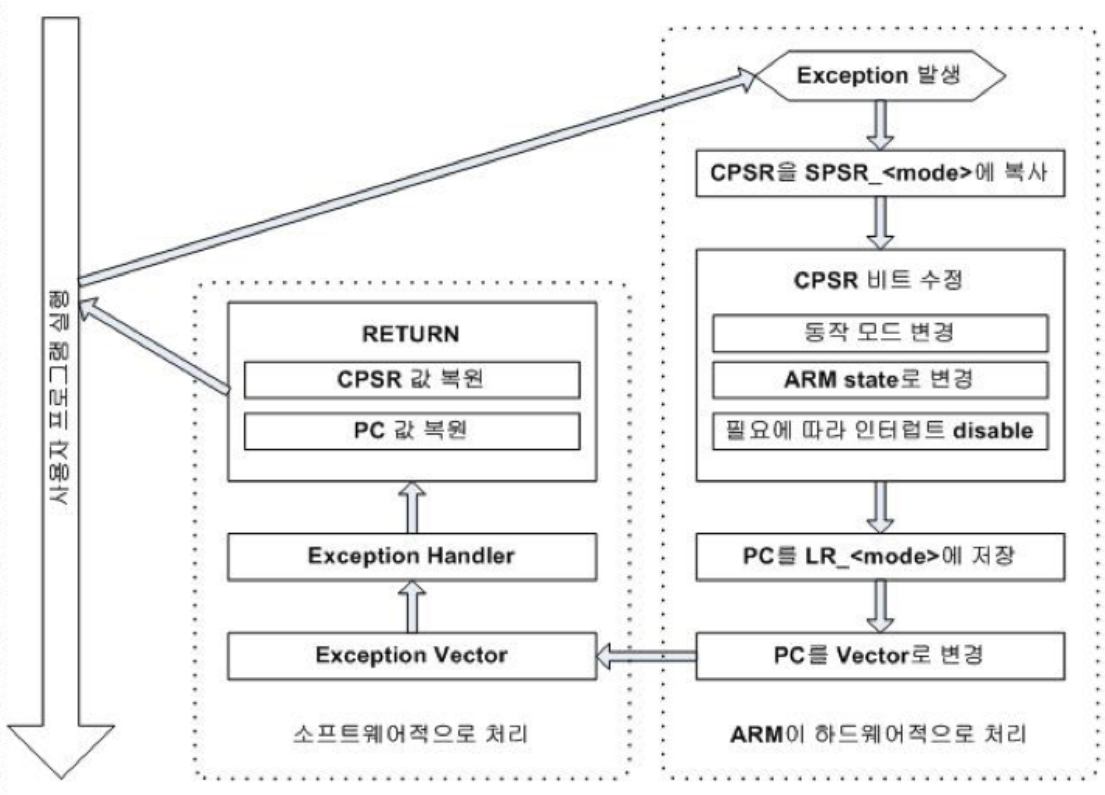
    Big-Endian : 낮은 주소에 상위 바이트 부터 기록

    Little-Endian : 낮은 주소에 하위 바이트 부터 기록

    왜 리틀 엔디안 사용? ALU에서 메모리를 읽는 방식이 메모리 주소가 낮은 쪽에서 부터 높은 쪽으로 읽어야 산술 연산의 수행이 더 쉽기 때문
  - Data access type
  - Aligned/Un-aligned access

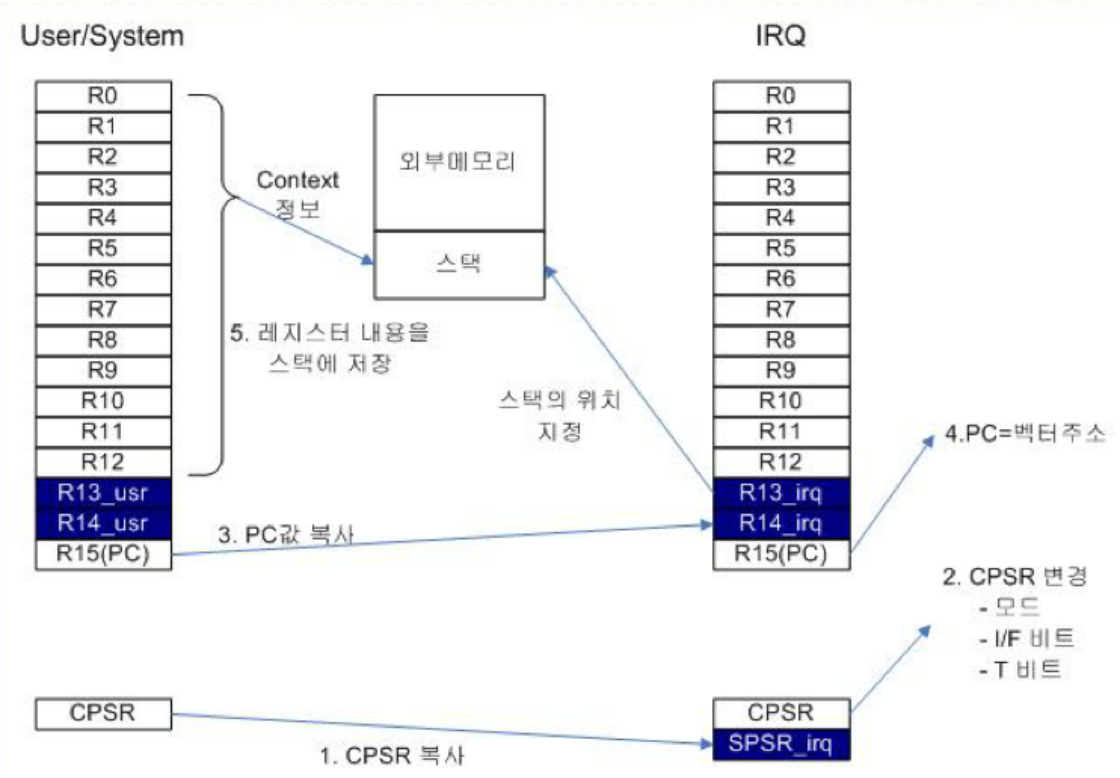
    정의 : 실행 속도를 높이기 위해 강제로 정렬하는 방식. 데이터와 명령어는 자신의 길이에 대한 배수 주소에 위치해야한다.

    왜 함? 하나의 명령어가 여러개의 위드에 걸쳐 자리잡을 일이 없기 때문에 메모리 읽기 사이클 수행 횟수를 최소화 할 수 있다.
- 예외처리
  - 정의 : 외부의 요청이나 오류에 의해서 정상적으로 진행되는 프로그램의 동작을 잠시 멈추고 프로세서의 동작 모드를 변환하고 미리 정해진 프로그램을 이용하여 외부의 요청이나 오류에 대한 처리를 하도록 하는 것
  - exacmple : Reset, Data Abort, FIQ, IRQ, Prefetch Abort, Undefined Instruction, Software interrupt
- exception vector (예외처리 백터) : 미리 정해진 프로그램의 위치를 칭함
- exception vector table : 각각의 exception에 대하여 vector를 정의해 놓은 테이블
  - 각 exception을 처리하기 위한 미리 정해진 주소
  - 메모리 안에서 고정된 공간
  - exception을 처리하기 위한 루틴으로 분기 할 수 있는 명령으로 구성
  - 기본 exception vector table 위치는 0x00000000
- Exception handling



1. 예외처리 발생
2. CPSR을 SPSR로 복사
3. CPSR 비트 수정
    1. Mode 수정
    2. ARM 모드로 변경 (T = 0)
    3. I/F 비트 수정
4. PC를 LR에 저장
5. PC를 vector로 변경
6. exception vector
7. exception handlr
8. pc값 복원
9. CPSR 복원

-

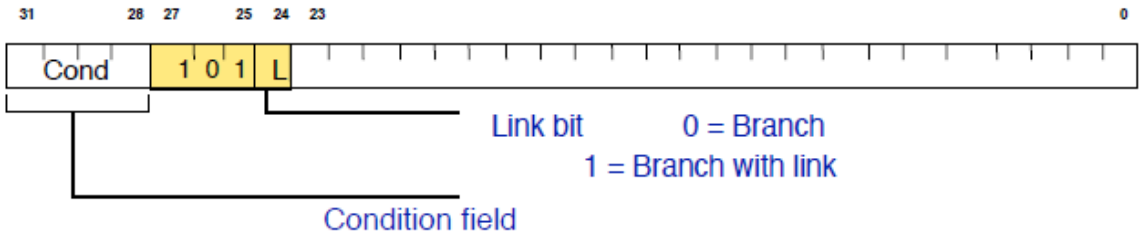


1. CPSR을 SPSR로 복사
2. CPSR 수정
    1. Mode 수정
    2. ARM 모드 수정
    3. I/F 수정
3. PC값 LR에 저장
4. PC를 exception vecor로 저장
5. 레지스터의 내용을 스택에 저장

- Conditional Execution
    - 모든 명령어들은 condition flag bits에 조건적으로 실행될 수 있음
    - pipeline stall를 야기시키는 branch cost를 상당히 줄일 수 있다.
    - CPSR와 비교 -> 맞아야지만 수행
    - 적절한 조건을 접미사로 붙여줌 ex. ADDS

      장점 : pipline이 crush되지 않고 execution을 optional하게 수행할 수 잇다.

7강



- PC값을 기준으로 비트 [23:0]에 해당하는 offset이 사용
- 24bit의 offset
    - 맨 상위비트는 +/- sign bit

- 나머지 23비트를 이용하여 2비트 왼쪽으로 shift한 값을 사용(word alignment - 하위비트 항상 00),

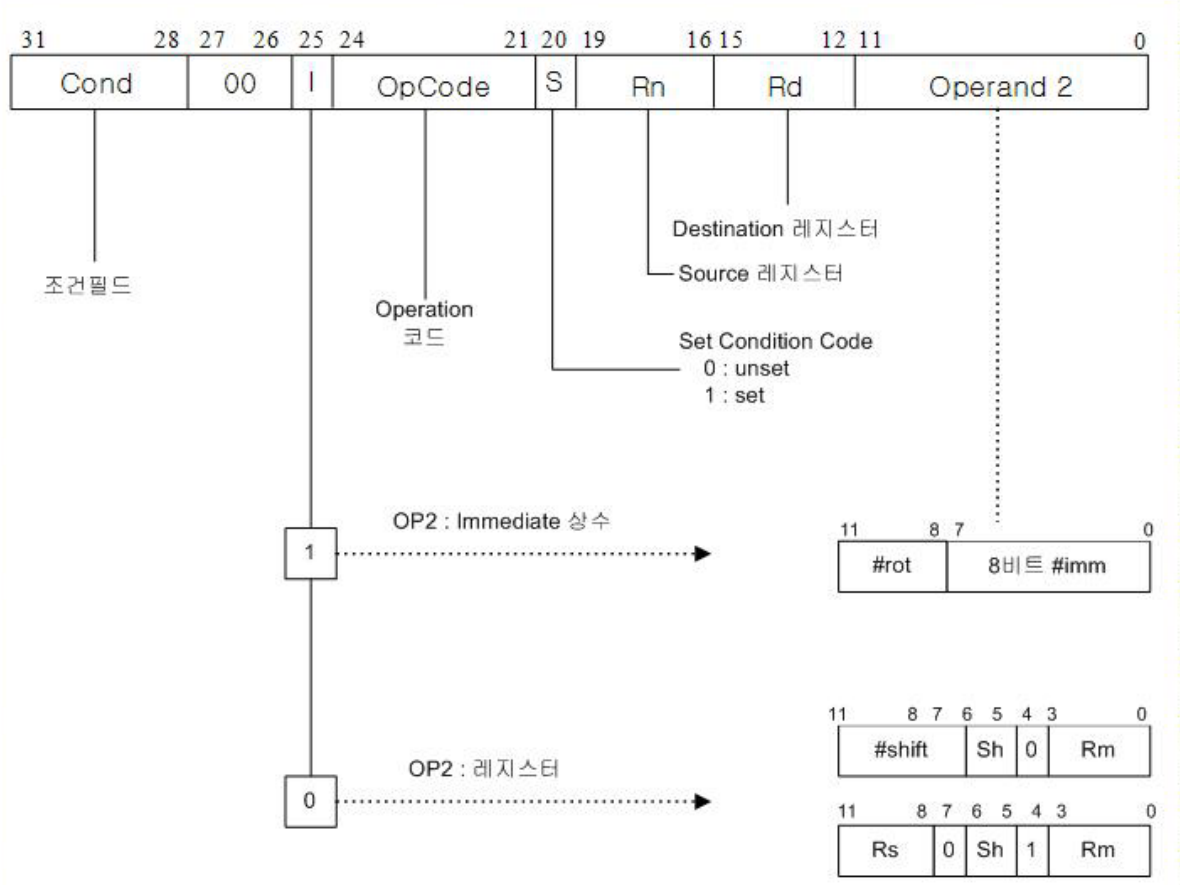
  실제상으로 32MB씩 위아래로 쓸 수 있음 ()

- bl 실행하면 LR에 PC-4가 들어가는 이유

```
                                        PC
base + 0 BL there         EX base+0
base + 4 add R0, R1, R2   ID base+4
base + 8 STR R0, [R4]     IF base+8 : LR <= base + 4 = (PC = base+8) - 4)
```

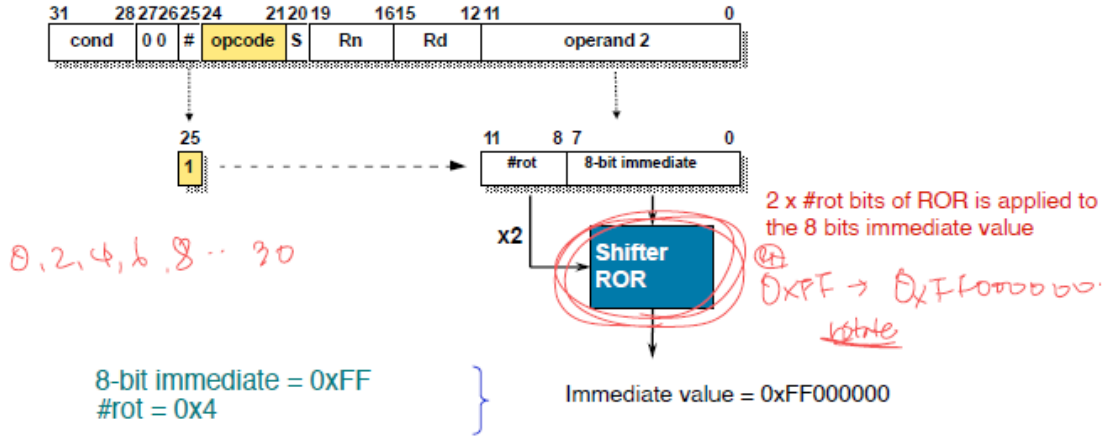
base + 0이 시작되고 있는 시점에서 base + 8 주소 명령어가 fetch되고, base + 4 주소 명령어가 decode 된다. PC란 fetch되는 주소를 가리키므로, base + 0를 시작하고 있을 때 PC값은 base + 8 이다. BL에서 돌아올 주소는 base + 8의 전 line인 base + 4이기 때문에 PC-4를 LR에 저장한다.

  - function 실행하고 돌아오면 조금 느려지는 현상 발생 : pipline이 깨져서 fetch랑 decode를 채워 넣는 추가 clock이 필요하기 때문에



- barrel shifter : 한 개의 연산으로 데이터 워드 내에 있는 다수의 비트를 이동하거나 회전시킬 수 있는 하드웨어장치

  ROR : 오른쪽 회전



- PSR(Program Status Register)
  - 유일한 접근 방법은 MRS, MSR을 통해 일반 레지스터로 옮기는 것
- CPSR
  - 모든 operation mode에서 접근할 수있음
  - user mode 에서만 조건필드가 수정될 수 있다
- SPSR
  - 모든 operation mode는 SPSR를 각각 가지고 있다. user mode 제외

8강

- ARM은 RISC 아키텍처
  - Load / Stroe 아키텍처를 의미
  - 메모리에서 직접 연산을 허용하지 않음 = 처리하고자 하는 데이터는 무조건 레지스터로 이동
    1. 데이터를 메모리에서 레지스터로 이동
    2. 레지스터에 읽혀진 값을 가지고 처리
    3. 연산 결과를 메모리에 저장
  - Load / Store 명령어 : LDR/STR, LDM/STM, SWP
- 단일 Load/Store

억세스 단위	Load 명령	Store 명령
워드(Word)	LDR	STR
바이트(Byte)	LDRB	STRB
하프워드(Halfword)	LDRH	STRH
Signed 바이트	LDRSB	
Signed 하프워드	LDRSH	

  - alignement 안맞으면 -> exception
  - 왜 signed가 따로 있을까? 그 자료형이 signed 로 선언되어있기 때문에, 산술 연산을 정상적으로 하려면 필요
    - unsigned :32bit에 맞춰서 zero extends
    - signed : 32bit에 맞춰서 sign extends