Program Status Regiser

- PSR Registers in ARM

  1 CPSR (Current Program Status Register)

  5 SPSR (Saved Program Status Register) (for each operating mode)

  CPSR을 복사해 넣는 특수 register, CPSR을 백업할때 씀

  언제 백업? mode를 바꾸게 되었을때

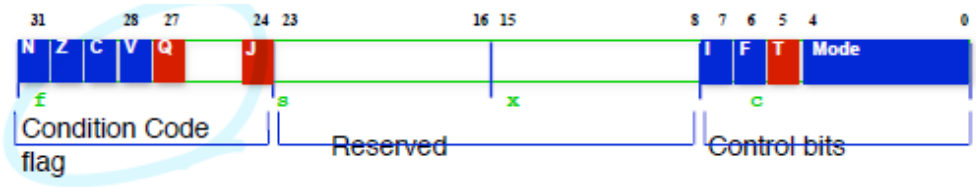
  R14: Linked Regitser 빵가루

  R13: STack Pointer

  R15: Program counter

  R0~R12: 저장용도

- PSR Register Fields
  - Condition flag
    - Reflects ALU processing results: 뭔가 연산한 후에 set되는 레지스터들
  - Control bits
    - To control CPU status



PSR Register - flag bits

- Flag bits는 ALU연산의 결과를 보여줌
  - N: 연산 결과가 마이너스면 1
  - Z: 연산 결과가 0이면 1
  - C(Carry): 연산 결과가 32bit를 넘으면(올림이 발생한 경우) 1
  - V(oVer flow): 연산결과가 32bit를 넘어 sign bit가 상실되면 1
- 추가 bits
  - Q (saturation bit)
  - J (Jazelle state)

PSR Register - control bits

- PSR control bits가 프로세서 작동 모드, 인터럽트, 프로세스 상태를 변경
  - I/F bit: Enable / Disable IRQ or FIQ
    - I: IRQ에 걸리는 걸 할 수 있게 / 없게(0: enable, 1: disable)
    - F: FIQ에 걸리는 걸 할 수 있게 / 없게(0: enable, 1: disable))
  - T bit
    - Tumb mode on / off : 32bit ~> 16bit
    - 'BX' 명령어를 통해서만 컨트롤됨
  - Mode bits (비트 안외워도 될 것 같음)

M[4:0]	Operating Mode	M[4:0]	Operating Mode
10000	User	10111	Abort
10001	FIQ	11011	Undefined
10010	IRQ	11111	System
10011	SVC		

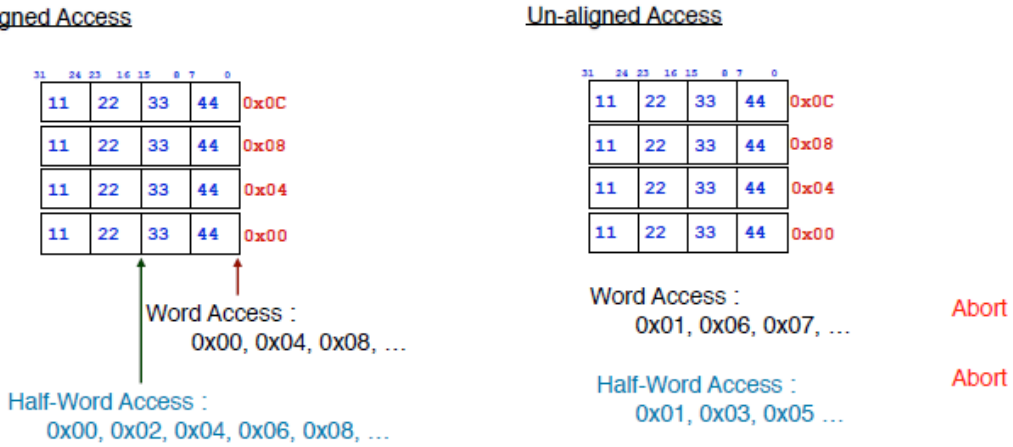
Memory Architecture

- Endian: 프로세서가 메모리에 저장하는 방식
  - Bing / Little endian (ARM은 둘다 됨)
  - 하드웨어 설정을 통해 구성됨
- Data 액세스 유형
  - Byte: 8bits
  - Halfword: 16 bits
  - Word: 32bits (32bit 프로세서 에서)
- Aligned / Un-aligned access : 메모리 내의 코드 또는 데이터의 배치를 워드 다누이 경계로 정렬
  - 모든 메모리 접근은 word(half-word)로 aligned(정렬)되어야함
  - 현대 아키텍처는 un-aligned(비정렬) 액세스 가능
    - 옛날 아키텍처는 항상 data 중단이 발생

    start address를 4의 배수로 안맞추면 할수는 있지만 exception 발생시킴 : abort

Aligned and Un-aligned Accesses

- 프로세서는 메모리 접근할 때 Byte, halfword(2byte), word(4byte) 단위로만 가능



Little-Endian vs. Big-Endian

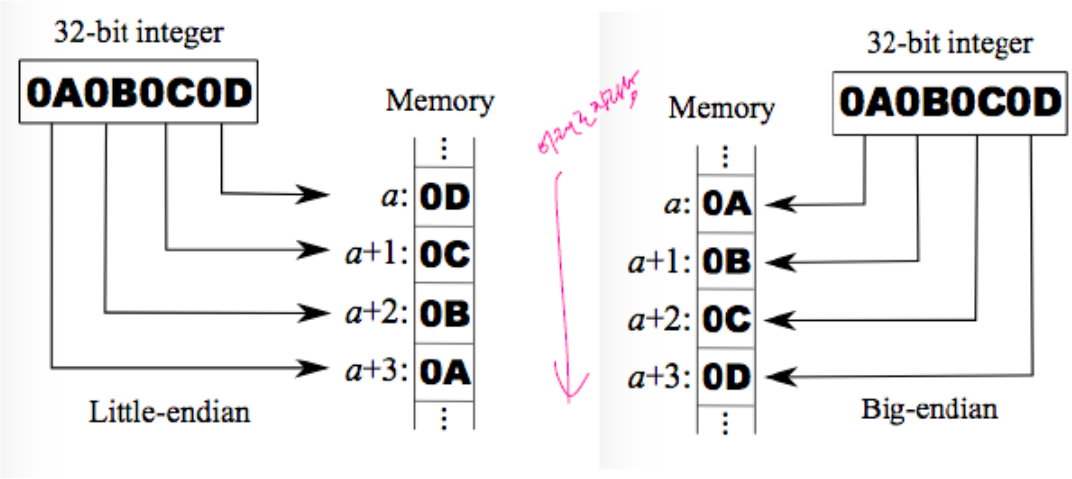
- Big-Endian: 큰 자릿수가 더 낮은 주소에 위치
- Little-Endian: 작은 자릿수가 더 낮은 주소에 위치

  12345678

  BIg: 12 34 56 78

  Little: 78 56 34 12

Little-Endian vs. Big.Endian



Exceptions

- 예외처리: 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
  - IRQ, FIQ를 비롯한 대부분의 Operating 모드는 외부에서 발새오디는 조건에 의해서 ARM 프로세서가 하드웨어적으로 변경
  - 외부에서 발생되는 물리적인 조건에 의해서 정상적인 프로그램의 실행을 미루고 예외적인 현상을 처리하는 것을 exception이라 한다.
  - operating 모드의 변경은 소프트웨어에 의하여 제어할 수 도 있다.
- ARM Exception
  - Reset
  - Data abort (데이터 읽어오기 불가)
  - FIQ (fast interrupt request)
  - IRQ (interrupt request)
  - Prefetch abort
  - Undefined instruction
  - Software interrupt

Exception Vectors

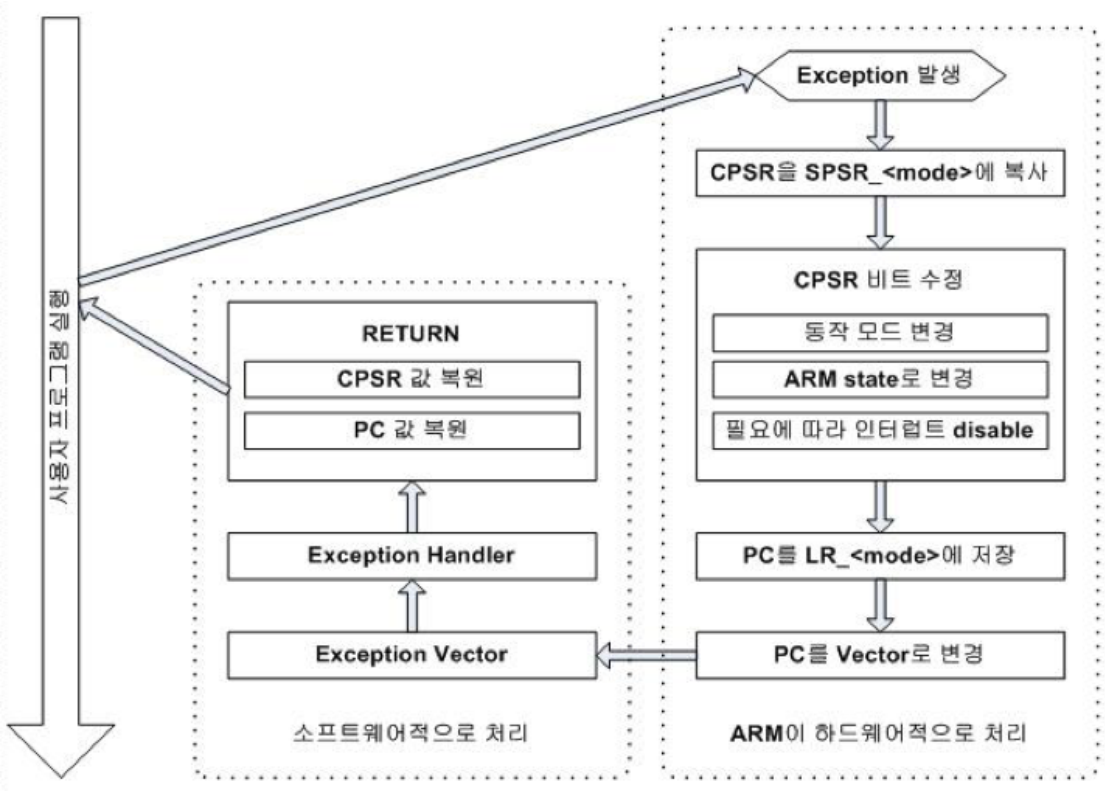
- Exception vector와 exception vector 테이블
  - **Exception vector**
    - exception이 발생하면 미리 정해진 address의 프로그램을 수행
    - 미리 정해진 프로그램의 위치를 excepetion vector라 한다.
  - **Exception vector table**
    - 발생 가능한 각각의 exception에 대하여 vector를 정의해 놓은 테이블
      - 각 exception 별로 1 word 크기의 명령어 저장 공간을 가진다
    - vector table에는 branch 또는 이와 유사한 명령으로 실제 exception을 처리하기 위한 루틴으로 분기할 수있는 명령으로 구성되어있다.
      - FIQ의 경우는 vector table의 맨 상위에 위치하여 분기 명령 없이 처리루틴을 프로그램할 수 있다.
    - ARM은 기본적으로 0x0000 0000에 vector table을 둔다. (MMU 제어 프로그램에 의해 위치 변경 가능)
  - 각 예외를 처리하기 위해 미리 결정된 주소
  - 모든 예외에 대한 예외 처리기 주소표
  - 메모리의 고정된 위치가 예외 벡터표에 할당됨

- 예외 벡터 테이블의 각 항목은 각 예외 핸들러에 대한 jump/branch 명령
- 기본 예외 백터 테이블 위치는 0x0000 0000

Exception Vector Table

Exception	Vector Address	Prioriry	Operation mode
Reset	0x0000 0000	1 (High)	Supervisor(SVC)
Undefined Instruction	0x0000 0004	6 (Low)	Undefined
Software Interrupt(SWI)	0x0000 0008	6	Supervisor(SVC)
Prefetch Abort	0x0000 000C	5	Abort
Data Abort	0x0000 0010	2	Abort
Reserved	0x0000 0014		
IRQ	0x0000 0018	4	IRQ
FIQ	0x0000 001C	3	FIQ

Exception Handling



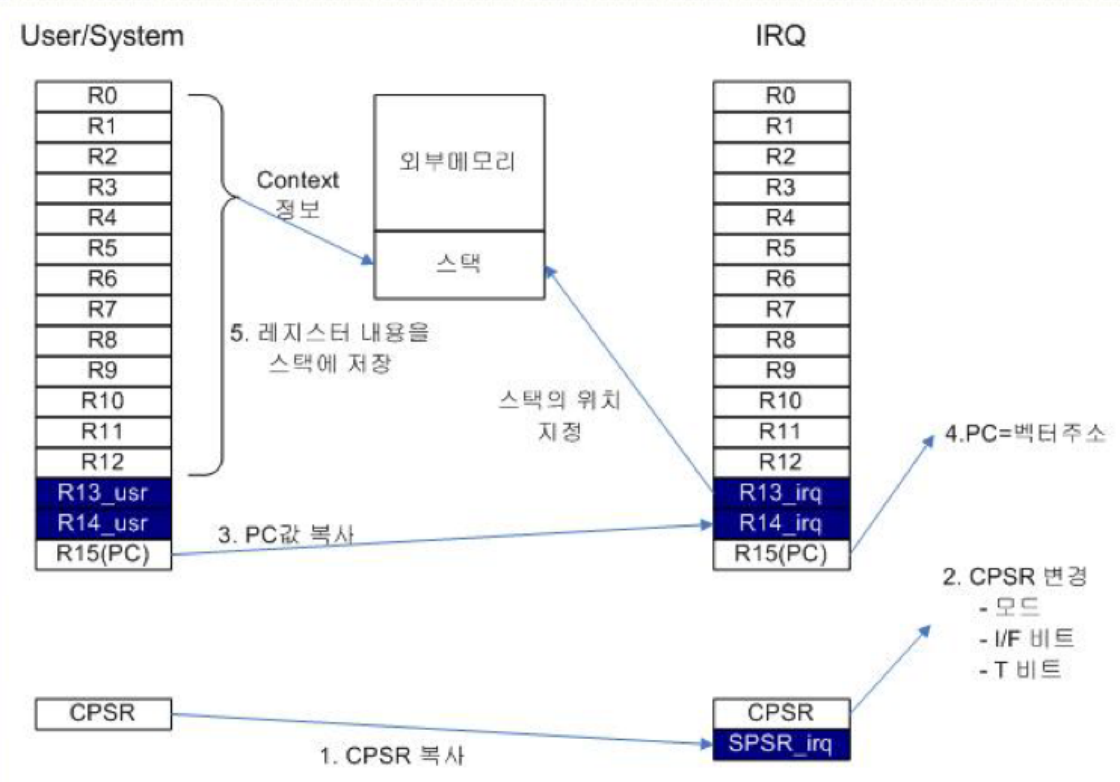
exception(FIQ든 interrupt든): hw/ sw 둘다 처리 해줍니다

- 예외처리 흐름
- 하드웨어 역할
    1. cpu가 정신을 잃음
    2. CPSR의 내용을 SPSR에 저장

3. CPSR의 값을 마음대로 바꿈
    1. mode에 관한 부분
    2. ARM 모드이냐 Thumb모드이냐 -> ARM모드로 무조건 바꿈 (T bit를 0 으로 설정)
    3. IRQ / FIQ enable/disable 바꿔줌
4. 현재 PC를 link register에 저장
5. vector address값을 pc에 설정

- 소프트웨어 역할
    1. Exception vector 처리
    2. Exception handler 처리
    3. PC 복구
    4. CPSR 복구

Exception Handling



- 예외처리 레지스터
- 하드위어적
    1. CPSR을 SPSR로 복사
    2. CPSR 변경
        1. 해당 모드로 변경
        2. I/F 비트 (인터럽트 막기)
        3. T 비트 (ARM 상태로 변경)
    3. PC값 저장
    4. PC을 exception vector로 변경
    5. 사용하던 현재 레지스터 내용을 스택(DRAM)으로 복사
    6. 스택의 위치를 저장

32 Bits ARM Instruction Format

Conditional Execution

- ARM에서 모든 명령을 조건(confidition flag bits)에 따라 실행 여부 결정 가능
- 분기 명령의 사용을 줄인다
  - 분기 명령이 사용되면 파이프라인이 스톨(stall)되고 새로운 명령을 읽어오는 사이클의 낭비가 따른다.
- 조건 필드 (Condition fields)
  - 모든 명령어는 조건 필드 (condition fields)를 가지고 있으며, CPU가 명령의 실행 여부를 결정하는데 사용된다.
  - 각 명령어의 조건 필드(condition field)를 CPSR의 condition flag와 비교한다.
  - 명령어는 조건 필드 (condition fields)와 CPSR의 조건 플래그(condition flag)가 일치할 때만 실행될 수 있다.

Conditional Execution

- 조건에 따라 명령어를 실행하도록 하기 위해서는 적절한 조건을 접미사로 붙여주면 됨:
  - 조건없이(Un-conditinal) 실행

    `ADD r0, r1, r2 ; r0 = r1 + r2 (ADDAL)`
  - Conditional 실행 : Zero flag가 set 되어 있을 때만 실행하고자 하는 경우

    `ADDEQ r0, r1, r2 ; if zero flag set`

    `then r0 = r1 + r2 (ADDAL)`
- data processing instruction에의 접미사 'S'가 없으면 CPSR의 조건 플래그(condition flags)에 영향을 미치지 않음

  `SUBS r0, r1, r2 ; r0 = r1 - r2`

  `...and set flags`
  - 데이터 처리 명령 중 비교를 위한 명령은 별도로 "S" 접미사를 붙이지 않아도 조건 플래그가 변경됨
- Data processing instruction : 레지스터 안에서 데이터를 조작하는데 사용
  - "S"접미사: 명령어 처리 후 결과를 가지고 cpsr 플래그들을 업데이트

ARM 명령어 요약

	Instruction Type	Instruction (명령)
1	Branch, Branch with Link	B, BL
2	Data Processing 명령	ADD, ADC, SUB, SBC, RSB, RSC, AND, ORR, BIC, MOV, MVN, CMP, CMN, TST, TEQ
3	Multiply 명령	MUL, MLA, SMULL, SMLAL, SMULL, UMLAL
4	Load/Store 명령	LDR, LDRB, LDRBT, LDRH, LDRSB, LDRSH, LDRT, STR, STRB, STRBT, STRH, STRT
5	Load/Store Multiple 명령	LDM, STM
6	Swap 명령	SWP, SWPB
7	Software Interrupt(SWI) 명령	SWI
8	PSR Transfer 명령	MRS, MSR
9	Coprocessor 명령	MRC, MCR, LDC, STC
10	Branch Exchange 명령	BX
11	Undefined 명령	

EQ

뒤에 s가 붙어있는거

얻을수 잇는 장점: pipline이 crush되지 않고 execution을 optional하게 수행할 수 있다.

Contidional Field