2️⃣

Chapter 2 (ARM)

Index

ARM modes

Thumb mode

- ARM mode의 반쪽짜리 버전이라 볼 수 있으며 16-bit instruction set

Supervisor	SVC
User	USR
System	SYS
Abort	ABT
Interrupt Request	IRQ
Fast Interrupt Request	FIQ
Undefined Instruction	UND

User : Normal Program execution mode

System : Run privileged operating system tasks

FIQ : When a high priority (fast) interrupt is raised

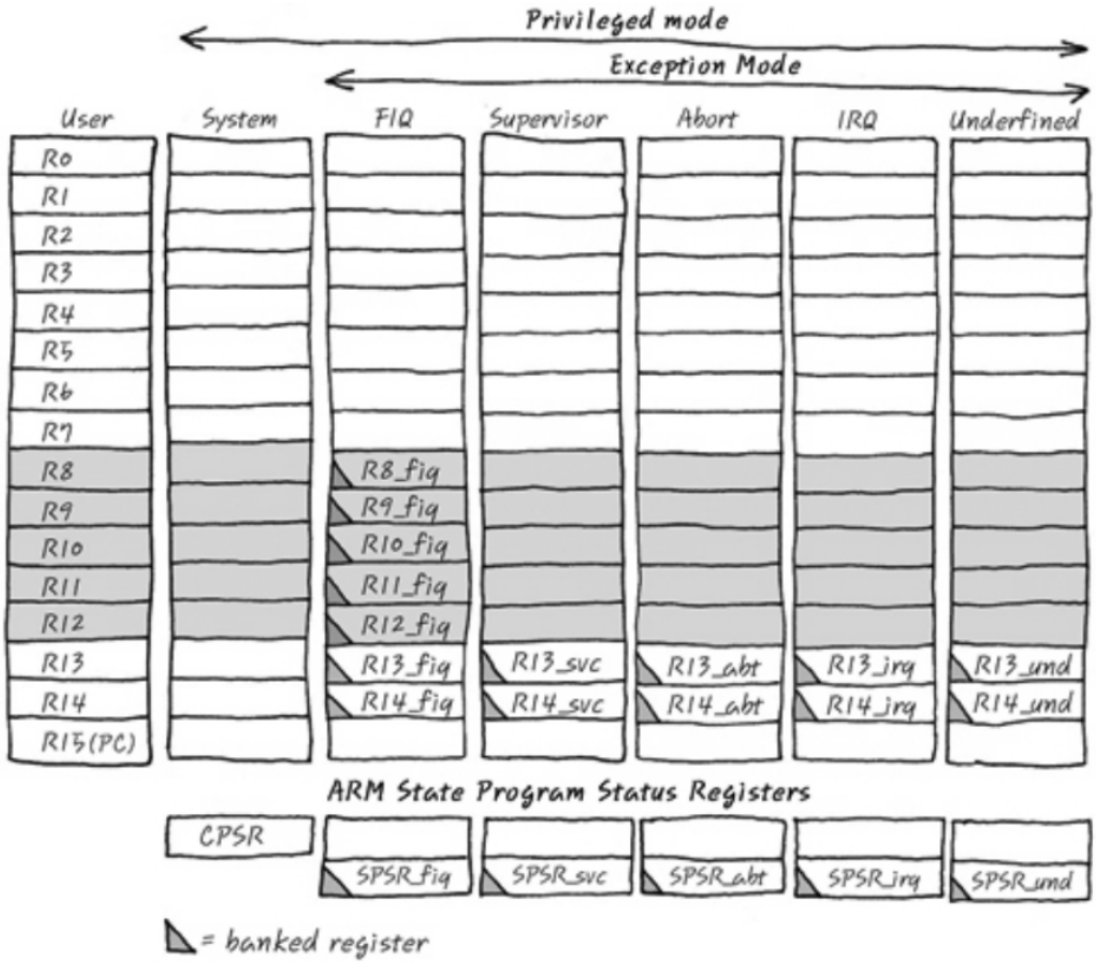
IRQ : When a low priority (normal) interrupt is raised

Supervisor : A protected mode for the operating system entered when a SWI instruction is executed

Abort : Used to handle memory access violations

Undef : Used to handle undefined instructions

▼ ARM register image



USR와 SYS는 같은 레지스터를 사용, 15개

FIQ는 자신만의 R8~R14 레지스터, 8개

FIQ가 레지스터가 많은 이유 : 빠른 연산 - 빠르게 인터럽트를 처리해야 하기 때문

SVC, ABT, IRQ, UND는 각각 자신만의 R13, R14, SPSR를 사용, 12개

+PC, CPSR

총 37개

CPSR

- Current PSR, 즉 현재 시스템의 상태를 나타내는 레지스터

SPSR

- Saved PSR, 즉 이전 시스템 상태를 백업하는 레지스터

context switching

- 특정 순간의 context를 저장/복원하면 그 특정 순간으로 돌아갈 수 있는 것

이때 각 프로세스 또는 함수가 어떤 레지스터를 사용해서 작업할지는 컴파일 단계에서 결정

ARM은 각 레지스터의 사용법에 대한 표준을 만들었고 이를 표준 버전에 따라 PCS, APCS, AAPCS

▼ Image

Register	Synonym	Special	Role in the procedure call standard
r15		PC	The Program Counter.
r14		LR	The Link Register.
r13		SP	The Stack Pointer.
r12		IP	The Intra-Procedure-call scratch register.
r11	v8	FP	ARM-state variable-register 8. ARM-state frame pointer.
r10	v7	SL	ARM-state variable-register 7. Stack Limit pointer in stack-checked variants.
r9	v6	SB	ARM-state v-register 6.
r8	v5		ARM-state variable-register 5.
r7	v4		Variable register (v-register) 4.
r6	v3		Variable register (v-register) 3.
r5	v2		Variable register (v-register) 2.
r4	v1		Variable register (v-register) 1.
r3	a4		Argument / result / scratch register 4.
r2	a3		Argument / result / scratch register 3.
r1	a2		Argument / result / scratch register 2.
r0	a1		Argument / result / scratch register 1.

R0~R3 : 배열로 넘길 때 사용, 4개 초과시 stack에 저장, scratch(연습장)레지스터라고도 불림, 중요한 정보를 넣어두면 안되고 넣어두었다면 스택에 넣고 복구하는 작업을 해줘야 한다.

R4~R11 : local 변수를 저장하는 용도, 한정된 개수를 넘어가게 되면 stack에 저장

R12~R15 : 특수용도 레지스터, R12-임시 저장소

> 모든 레지스터는 사용하기 전에 기존에 들어있는 값을 stack에 저장한 뒤 사용해야 하고, 서브루틴 끝난 후 복귀할 때는 다시 stack에서 꺼내 복귀해줘야 한다.

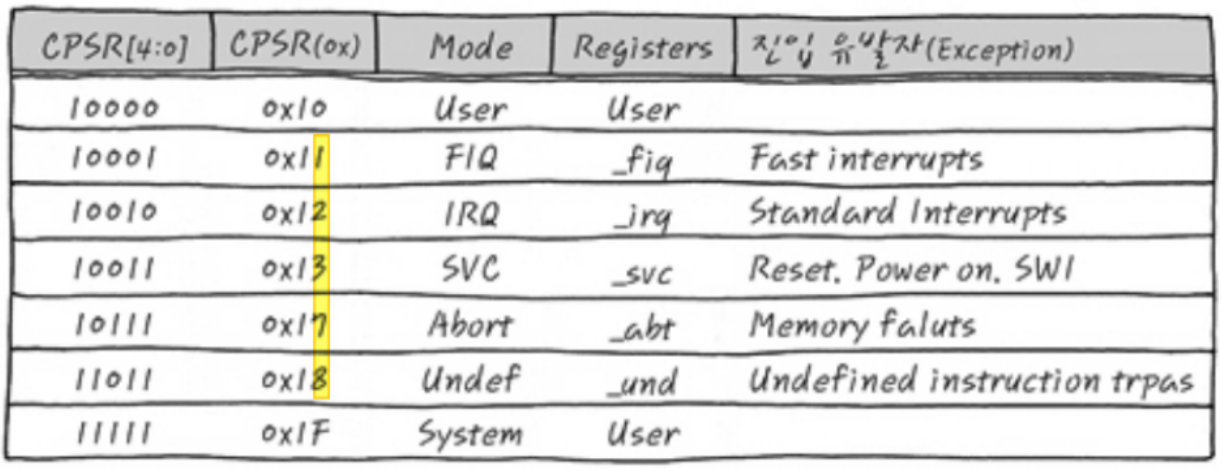
Exception

- interrupt를 포함한 큰 사건(외부 요청, 내보 오류 등)을 의미

exception발생 시 진행중인 동작을 멈추고 priviledged mode로 진입한 뒤 exception handler의 주소로 jump해 처리

Exception Vector Table

- exception이 발생했을 때 jump하는 address들을 모은 것



CPSR[4:0]	CPSR(0x)	Mode	Registers	진입 유발자(Exception)
10000	0x10	User	User	
10001	0x11	FIQ	_fiq	Fast interrupts
10010	0x12	IRQ	_irq	Standard Interrupts
10011	0x13	SVC	_svc	Reset, Power on, SWI
10111	0x17	Abort	_abt	Memory faluts
11011	0x18	Undef	_und	Undefined instruction trpas
11111	0x1F	System	User	

SVC mode : Power-on & Reset

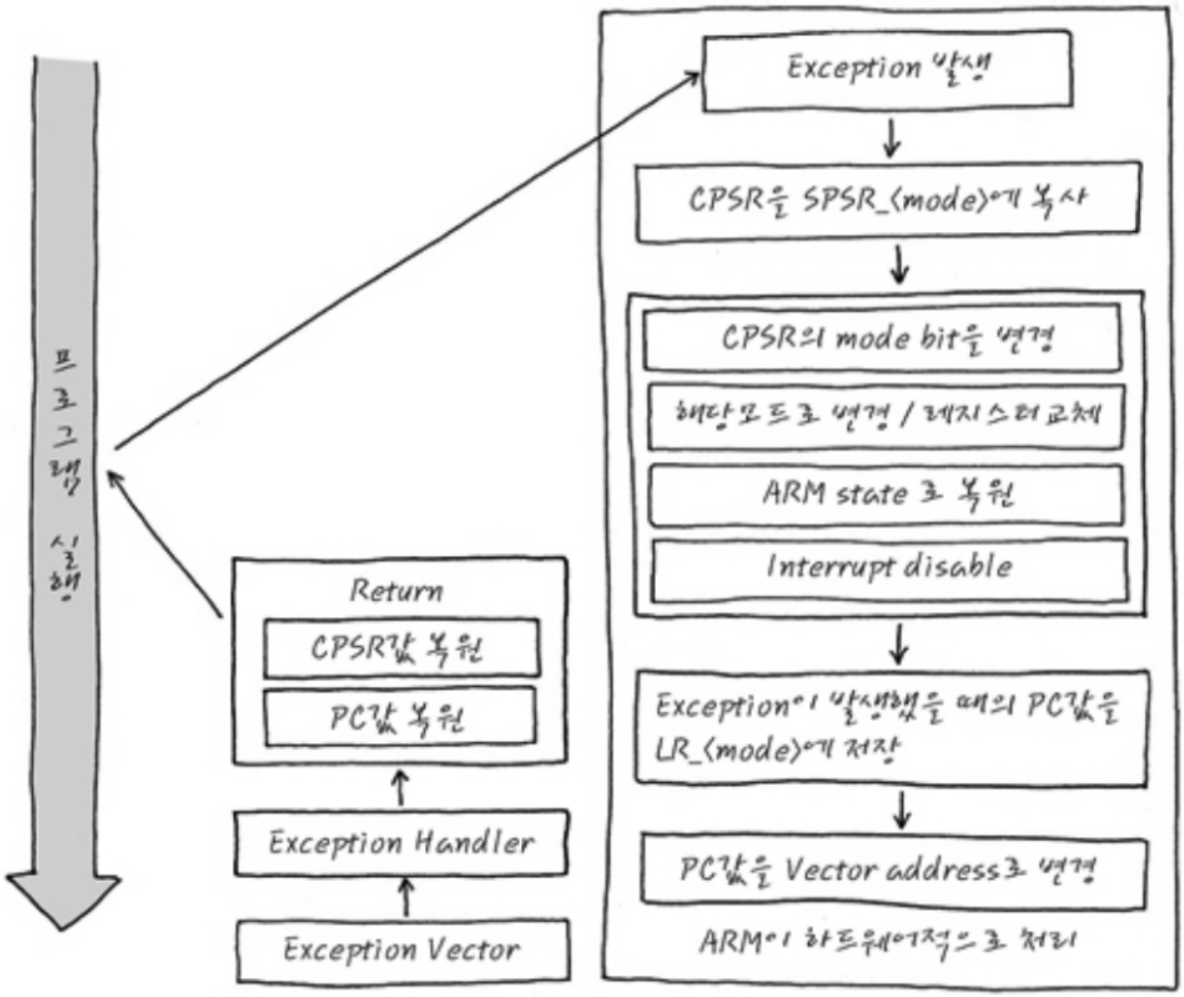
IRQ mode : HW interrupt

FIQ mode : FIQ에 등록된 HW interrupt

ABT mode : 접근할 수 없는 주소로의 접근시도

UND mode : 이해할 수 없는 명령어(opcode)

“노란색으로 표시한 hex값을 이용해 외어두면 좋다.”



- 애플리케이션 실행 중 exception이 일어났다면?
  1. 현재 진행 중인 애플리케이션에 대한 context를 백업
     a. SPSR에 CPSR값 저장
     b. R13 움직여서 R0~R12, R14값 저장
     c. R14에 실행중이던 line 주소 저장 (pipeline에 의해 현재 exception이 발생한 주소와 PC값은 2개 명령어 차이가 나기 때문에 저장,복귀할 때 **주소 보정 작업 필요**, 그 뒤 주소로 이동해서 실행 가능)
  2. PC를 exception vector table 시작 주소로 바꾸고 jump
- 애플리케이션 실행 중 interrupt가 발생했다면?

1. MCU의 interrupt controller라는 HW로 pin을 통해 전기 신호를 주면 ARM core가 알아챈다.
2. IRQ mode로 진입, vector table → IRQ handler로 jump해서 대응하는 ISR을 확인
3. ISR로 jump, interrupt를 처리 (ISR은 간결하고 짧게)

자신만의 register set을 갖고 있으면 mode 전환이 빠르고 보안상 장점도 있다.

USR과 SYS에는 SPSR이 없다

privileged mode에서 CPSR의 하위 5-bit를 바꿔서 USR모드로 진입 가능

ARM SoC

ARM

- processor를 만들어서 파는 회사가 아니라 CPU architecture를 파는 회사
- design한 architeture를 가져다 쓰는 회사로부터 로열티를 받아 비지니스를 하는 회사

SoC

- core에 여러 기능을 덧붙혀 하나의 chip으로 만든 것
- IP : SoC 내부의 각각 기능을 가진 block

AMBA (Advanced Microcontroller Bus Architeture) 프로토콜

- ARM은 내부 IP들끼리 잘 통신하면서 core를 효율적으로 활용한 방법을 적용한 Bus 프로토콜

Bus 프로토콜 : bus 위로 data를 어떻게 송수신 할 것인지에 대한 약속

AHB(AMBA : High performance)

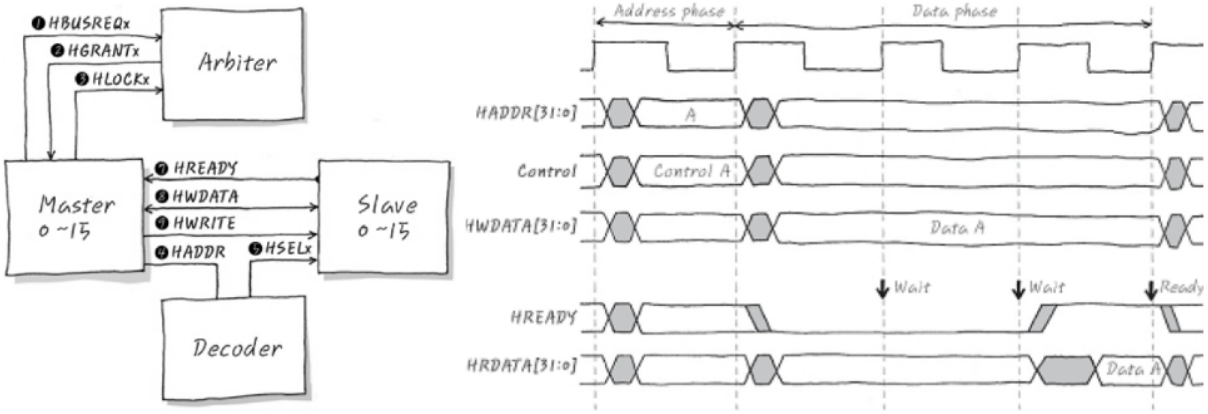
- 이름대로 ‘burst mode’를 지원해 data의 빠른 전송이 가능한 고성능 bus 인터페이스

APB(AMBA : Peripheral)

- AHB는 고속 bus, 저속인 peripheral IP가 중간에 끼면 전체 속도가 느려진다. 저속 IP들을 모아 bus 인터페이스를 만들어 효율성 up, Mux 기반 bus이므로 address, control, data line이 모두 하나의 bus로 통합해 시분할로 운용

ASB(AMBA : system)

아비터 - Bus를 사용할 때 중재 역할



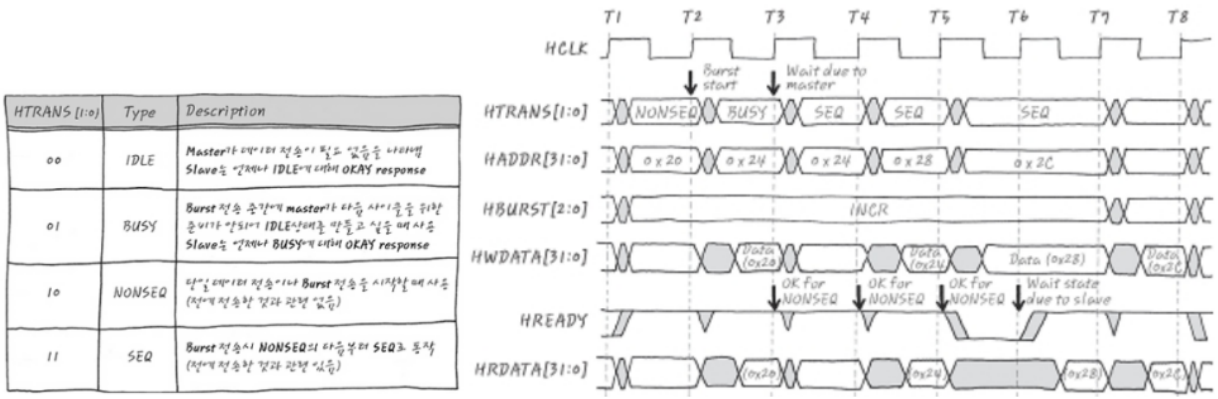
1. Master는 arbiter에게 bus를 쓰고 싶다고 HBUSREQx 신호를 보낸다.
2. Arbiter는 다른 bus를 쓰고 싶다면 읽씹하고, 쓰고 싶지 않다면 HGRANTx 신호로 대답
3. Master는 이제 쓴다고 HLOCKx 신호를 arbiter에게 보낸다. arbiter는 master보다 priority가 낮은 IP에게는 bus 사용을 허가 X
4. Master는 원하는 slave의 address인 HADDR을 decoder에게 준다. Decoder는 해당 slave에게 HSELx 신호를 줘서 접근을 원한다는 걸 알려 활성화
5. Master는 원하는 slave에게 control 신호인 HWRITE을 전달
6. Slave는 준비가 끝났다고 HREADY로 대답
7. Master는 HWDATA신호에 write하고 싶은 data를 써서 slave에게 전달하거나 Slave는 master가 요청한 data를 read하도록 master에게 전달 (양방향)

Transaction(트랜젝션) 과정은 address phase와 data phase로 나뉨

Address phase 때 master가 HADDR과 HWRITE을 전달

이후 2번의 clock cycle동안 준비 (wait), master는 slave가 읽을 때까지 계속 HWDATA를 유지

준비가 끝난 slave가 HREADY를 보내면 그때 master의 HWDATA를 인식한 뒤 write할 것 또는 이때 준비한 HDATA를 master에게 전달



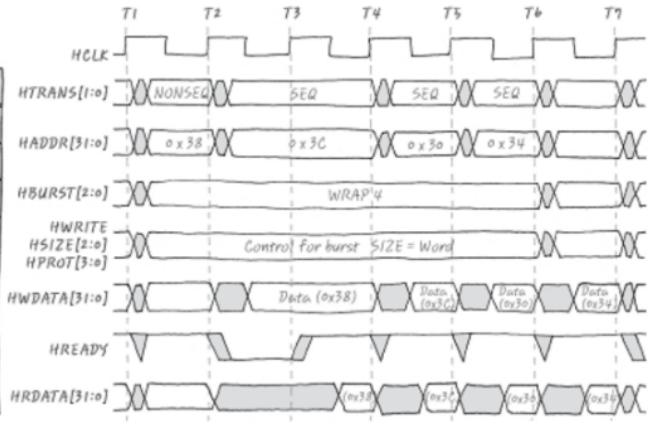
HTRANS [1:0]	Type	Description
00	IDLE	Master가 데이터 전송이 필요 없음을 나타냄 Slave는 언제나 IDLE에 대해 OKAY response
01	BUSY	Burst 전송 중간에 master가 다음 사이클을 위한 준비가 안되어 IDLE상태로 만들고 싶을 때 사용 Slave는 언제나 BUSY에 대해 OKAY response
10	NONSEQ	단일 데이터 전송이나 Burst 전송을 시작할 때 사용 (전에 전송한 것과 관련 없음)
11	SEQ	Burst 전송시 NONSEQ의 다음부터 SEQ로 동작 (전에 전송한 것과 관련 있음)

- **HTRANS** : 트랙잭션의 mode를 알려주는 역할, burst mode를 지원하는 AHB의 경우 현재 데이터와 다음 데이터 사이의 연관성을 나타내는 역할

1. HADDR이 0x20을 가리키며 HTRANS에 NOSEQ를 날려 새롭게 데이터를 전송 시작한다고 전달
2. HREADY로 대답한 slave는 1-cycle 뒤에 (0x20에 대한) HWDATA를 받거나 HRDATA를 제공
3. master에 문제가 생겨 다음 전송을 할 수 없게 됐다. 1-cycle만 쉬기 위해 HTRANS에 BUSY를 날리면, slave는 알겠다고 대답
4. 이제 전송이 가능, HADDR에 0x24를 가리키며 HTRANS에 SEQ을 날린다. 즉 brust mode를 사용 중이고 데이가 연관돼있다고 알려주는 것

- brust mode를 사용하면 한 번의 address phase 이후 연속해서 data 접근
- address phase 생략위해, HADDR이 자동으로 증가하거나 감소, HBURST 신호를 통해 증가하거나 감소하는 bit수 조절

HBURST [2:0]	Type	Description
000	SINGLE	Single transfer
001	INCR	Incrementing burst of unspecified length
010	WRAP 4	4-beat wrapping burst
011	INCR 4	4-beat increment burst
100	WRAP 8	8-beat wrapping burst
101	INCR 8	8-beat increment burst
110	WRAP 16	16-beat wrapping burst
111	INCR 16	16-beat increment burst



- HBURST는 3-bit로 다양한 brust mode를 표현
  - INCR x : 이름 그대로 x-bit씩 주소가 증가하는 것
  - WRAP x : 주소가 wrapping 돼 boundary 내부에서만 burst 한다는 뜻
    - WRAP 뒤에 붙은 정수 x가 Word 단위의 address boundary
    - WRAP 4 라면? - word:4 , 1word=4byte , 16byte=0x10 , 0x10단위로 boundary
- AMBA가 발전하면서 **AXI**도 나왔다.
  - Brust mode 기반 bus, 시작주소만으로도 burst transfer 가능
  - Response channel이 추가
  - R/W가 동시 가능
  - ARM11 이상의 backbone bus로 이용