제2장 프로그래머의 모델

개요

S3C2410X는 Advanced RISC Machines, Ltd에서 개발한 성능이 향상된 ARM920T 코어를 내장하고 있다.

프로세서 동작 상태

프로그래머의 관점에서 보면, ARM920T는 아래와 같은 2가지 상태 중 하나에 있게 된다:

☞ 32-bit, word-단위의 ARM 명령어를 실행할 수 있는 ARM 상태.

☞ 16-bit, halfword-단위의 THUMB 명령어를 실행할 수 있는 THUMB 상태. 이 상태에서, PC는 halfword 단위를 변경 및 선택하는데 bit 1을 이용한다.

주의할 점

이러한 2가지 상태의 상호변경은 프로세서의 모드나 레지스터의 내용에 영향을 미치지는 않는다.

상태의 변환

THUMB 상태로의 진입

THUMB 상태로의 진입은 오퍼랜드 레지스터 안의 상태 bit(bit 0)을 set하는 BX 명령어를 실행하면 된다. 프로세서가 THUMB 상태에 있을 때, exception이 발생하면, exception(IRQ, FIQ, UNDEF, ABORT, SWI 등.)에서 복귀할 때 THUMB 상태로의 변경이 자동적으로 이루어진다.

ARM 상태로의 진입

아래와 같은 상황을 통해서 ARM 상태로의 진입이 발생한다:

☞ 오퍼랜드 레지스터의 상태 bit를 clear 하는 BX 명령어를 실행

☞ 프로세서가 exception(IRQ, FIQ, RESET, UNDEF, ABORT, SWI 등)을 취함. 이러한 경우에, PC는 exception 모드를 갖는 링크 레지스터에 놓이게 되며, exception의 벡터 어드레스에서 실행이 시작된다.

메모리 포맷

ARM920T는 메모리를 0byte에서 상위 byte의 선형 배열 모음으로 간주한다. byte0에서 byte 3까지는 첫 번째 word에 저장되고, byte 4에서 byte 7까지는 두 번째 word에 저장된다. ARM920T는 메모리 안의 word를 Big-Endian 혹은 Little-Endian 형식으로 다룬다.

Big-Endian 형식

Big-Endian 형식에서, word의 MSB는 하위 byte에 저장되고, LSB는 상위 byte에 저장된다. 이러한 이유로, 메모리의 byte 0은 24에서 31라인 까지를 가리킨다.

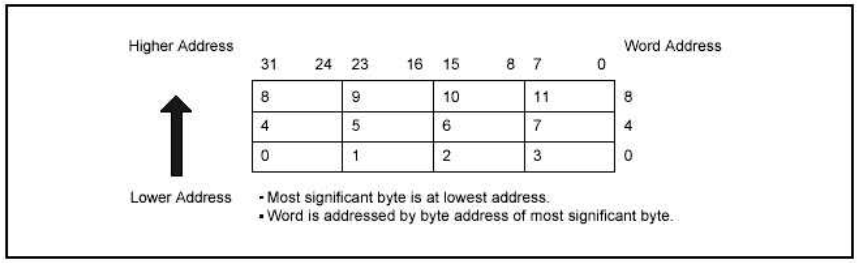

그림 2-1. Byte 단위의 Big-Endian 어드레스

Little-Endian 형식

Little-Endian 형식에서, Word의 하위 byte는 Word의 LSB가 되고, 상위 byte는 Word의 MSB가 된다. 이리하여, 메모리의 byte 0은 데이터 라인 0에서 7까지 이다.

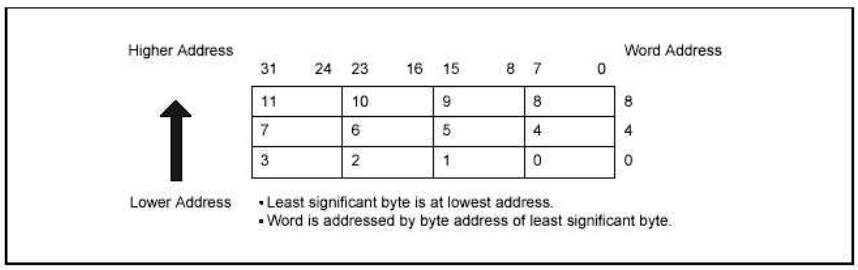

그림 2-2. Byte 단위의 Little-Endian 어드레스

명령어 길이

명령어의 길이는 ARM 상태에서는 32 bit, THUMB 상태에서는 16 bit이다.

데이터의 형태

ARM920T는 byte(8-bit), halfword(16-bit), word(32-bit)의 데이터 형식을 지원한다. Word는 4-byte씩, half word는 2-byte씩 나뉘어 배열된다.

동작 모드

ARM920T는 아래와 같이 7가지의 동작 모드를 지원한다:

☞ User(usr) 모드 : 일반적인 ARM 프로그램 수행 상태
☞ FIQ(fiq) 모드 : 데이터 전송 혹은 채널 처리을 지원
☞ IRQ(irq) 모드 : 범용의 인터럽트를 핸들링 하는데 사용
☞ Supervisor(svc) 모드 : OS 보호 모드
☞ Abort mode(abt) 모드 : 데이터 혹은 명령어의 prefetch abort 후에 진입하는 모드
☞ System(sys) 모드 : OS를 위한 특권 사용자 모드
☞ Undefined(und) 모드 : 정의되지 않은 명령어가 수행될 때 진입하는 모드

위의 모드 간의 전환은 소프트웨어 제어로 할 수도 있고, 외부 인터럽트나 exception 수행을 통해서도 이루어질 수 있다. 대부분의 응용 프로그램은 사용자 모드에서 수행된다. 특권 모드로 알려진 비-사용자 모드는 인터럽트 서비스나 exception, 혹은 보호 자원에 접근하기

위해서 진입하는 모드이다.

레지스터

ARM920T는 31개의 범용 32-bit 레지스터와 6개의 상태 레지스터를 합한 37개의 레지스터를 갖는데, 한 번에 37개 모두를 사용할 수는 없다. 프로세서의 상태와 동작 모드는 프로그래머에게 어느 레지스터가 유용한지에 대한 정보를 제공한다.

ARM 상태의 레지스터 set

ARM 상태에서는, 한 번에 16개의 범용 레지스터와 1개 혹은 2개의 상태 레지스터 사용이 가능하다. 특권 모드(비-사용자 모드)에서는, 특수 뱅크 레지스터로 변환된다. 그림 2-3은 각 모드에서 어느 레지스터가 사용이 가능한지를 나타내고 있다: 뱅크 레지스터는 음영의 삼각형으로 표시를 하였다.

ARM 상태 레지스터는 직접 접근이 가능한 16개의 레지스터를 가지고 있다: R0에서 R15까지 이다. R15를 제외한 모든 레지스터는 범용이며, 데이터나 어드레스의 값을 홀딩(hold)하는데 사용될 수도 있다. 이 외에 추가로, 상태 정보를 저장하는데 사용되는 17번째 레지스터가 존재한다.

레지스터 14 : 서브루틴 링크 레지스터로 사용된다. Branch & Link(BL) 명령어가 수행될 때 R15의 내용을 복사한다. 이 경우를 제외하고는 항상 범용 레지스터로 사용된다. 인터럽트나 exception이 발생할 때, 혹은 Branch & Link 명령어가 인터럽트 혹은 exception 루틴 안에서 실행될 때, 뱅크 레지스터 R14_svc, R4_irq, R14_fiq, R14_abt와 R14_und는 R15의 리턴 값을 홀딩하는데 사용된다.

레지스터 15 : 프로그램 카운터(PC)를 홀딩한다. ARM 상태에서, R15의 bit [1:0]은 0이고 bit [32:2]는 PC를 포함한다. THUMB 상태에서는, bit [0]은 0이며, bit [31:1]은 PC를 포함한다.

레지스터 16 : CPSR(현재의 프로그램 상태 레지스터)이다. 이 레지스터는 상태 코드 플래그와 현재의 모드 bit를 포함한다.

FIQ 모드는 R8-14(R8_fiq - R14_fiq)로 맵핑되는 7개의 뱅크 레지스터를 갖는다. ARM 상태에서, 많은 FIQ 핸들러는 어떤 레지스터도 저장할 필요가 없다. User, IRQ, Supervisor, Abort와 Undefined 는 각각 R13과 R14에 맵핑되는 2개의 뱅크 레지스터를 갖으며, 이러한 모드는 각각 개별적인 스택 포인터와 링크 레지스터를 갖는다.

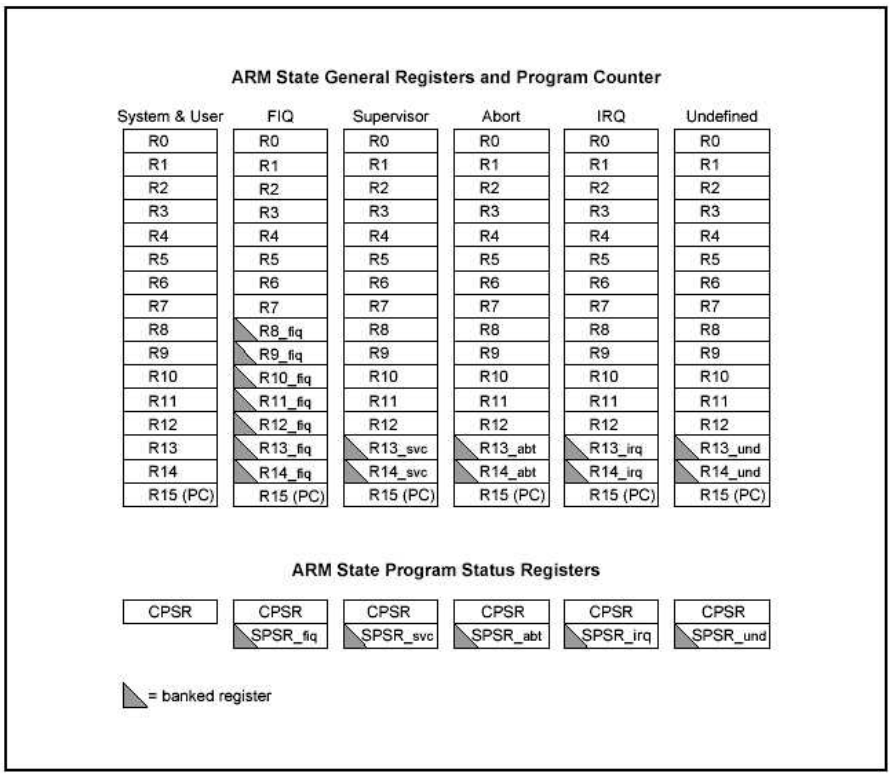

그림 2-3. ARM 상태의 레지스터 구조

THUMB 상태의 레지스터 set

THUMB 상태의 레지스터 set은 ARM 상태 set의 부분집합이다. 프로그래머는 8개의 범용 레지스터인 R0 – R7, 프로그램 카운터(PC), 스택 포인터 레지스터(SP), 링크 레지스터(LR), CPSR에 직접 접근이 가능하다. 각 특권 모드에는 뱅크 스택 포인터, 링크 레지스터, 저장된 프로세서 상태 레지스터(SPSR)가 있다. 그림 2-4를 참조 하시오.

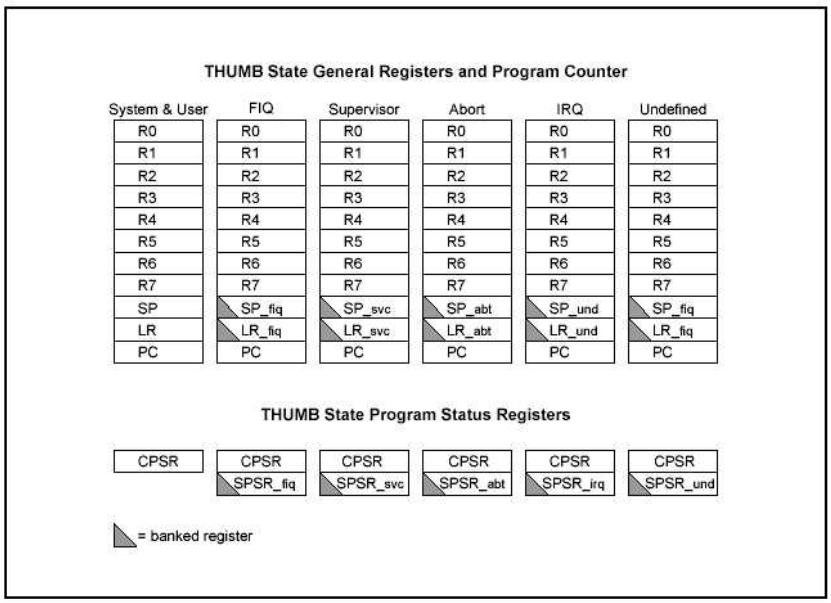

그림 2-4. THUMB 상태의 레지스터 구조

ARM 상태 레지스터와 THUMB 상태 레지스터 사이의 관계

THUMB 상태 레지스터는 아래와 같이 ARM 상태 레지스터와 관련이 있다:

☞ THUMB 상태의 R0 - R7 레지스터와 ARM 상태의 R0 - R7 레지스터는 같음

☞ THUMB 상태의 CPSR, SPSR과 ARM 상태의 CPSR, SPSR은 같음

☞ THUMB 상태의 SP는 ARM 상태의 R13 레지스터로 맵핑됨

☞ THUMB 상태의 LR은 ARM 상태의 R14 레지스터로 맵핑됨

☞ THUMB 상태의 프로그램 카운터는 ARM 상태의 프로그램 카운터(R15)로 맵핑됨

위의 관계는 그림 2-5에 나타나 있다.

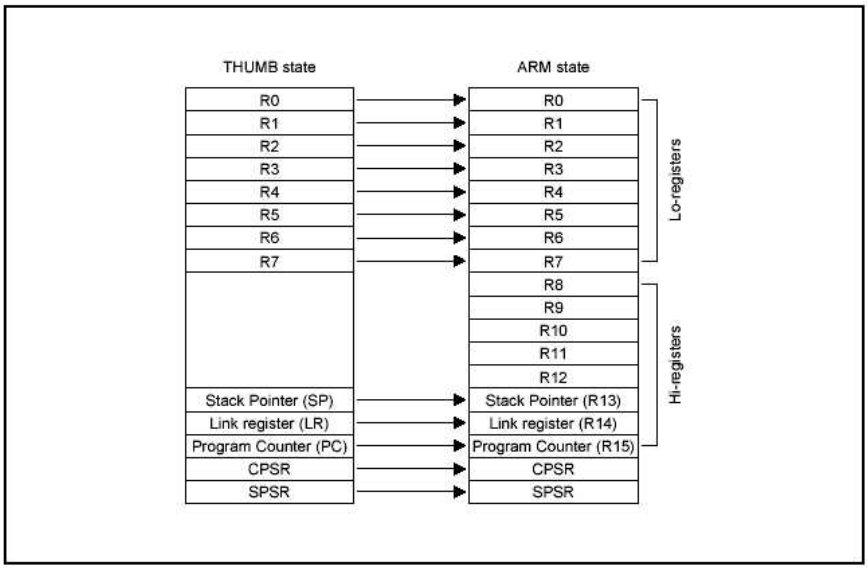

그림 2-5. THUMB 상태에서 ARM 상태로의 레지스터 맵핑

THUMB 상태의 상위 레지스터에 접근

THUMB 상태에서, R8-R15 레지스터(상위 레지스터)는 표준 레지스터에 속하지 않는다. 어쨌든, 어셈블리 언어 프로그래머는 이 레지스터들에 접근 시 제한이 있으며, fast temporary storage에서 사용할 수는 있다.

MOV 명령어의 특수 변수를 이용하면, R0-R7 레지스터(하위 레지스터)에서 상위 레지스터로, 혹은 상위 레지스터에서 하위 레지스터로 값을 전송할 수 있다. 상위 레지스터의 값은 CMP와 ADD 명령어를 이용해서 하위 레지스터 값에 더하거나 2개의 값을 비교할 수도 있다. 그림 3-34에 자세한 내용이 나와 있다.

프로그램 상태 레지스터

ARM920T는 현재의 프로그램 상태 레지스터(CPSR)와 exception 핸들러가 사용하는 5개의 프로그램 저장 상태 레지스터(SPSRs)를 가지고 있다. 이이 레지스터의 기능은 아래와 같다:

☞ 가장 최근에 수행된 ALU 동작에 관한 정보를 가지고 있음
☞ 인터럽트의 인에이블/디스에이블 컨트롤
☞ 프로세서의 동작 모드를 설정

bit의 배열이 그림 2-6에 나타나 있다.

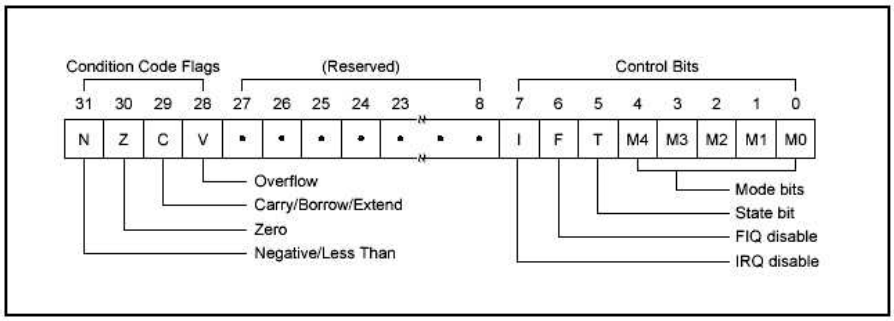

그림 2-6. 프로그램 상태 레지스터 포맷

상태 코드 플래그

N, Z, C, V bit는 상태 코드 플래그이다. 이 플래그들은 연산이나 로직 동작으로 인해서 바뀔 수 있으며, 명령어가 수행될지를 결정하는데 테스트 용으로 사용될 수도 있다.

ARM 상태에서, 모든 명령어는 상태에 따라서 수행된다: 표 3-2를 보시오.

THUMB 상태에서는, 오직 Branch 명령어만이 상태에 따른 명령어 수행이 가능하다: 그림 3-46을 보시오.

컨트롤 bit

PSR(I, F, T, M[4:0])의 하위 8 bit는 각 모드를 컨트롤 하는 bit 이다. 이러한 값들은 exception이 발생할 때 변경될 수 있다. 프로세서가 특권 모드에서 동작하고 있으면, 이들은 소프트웨어적으로 조작이 가능하다.

T bit : 이 bit는 동작 상태를 나타낸다. 이 bit가 1로 설정되어 있을 때, 프로세서는 THUMB 상태에서 실행되지만, 그 외에는 ARM 상태에서 실행된다. 이러한 상태는 외부신호 TBIT에 영향을 준다. 소프트웨어적으로는 CPSR의 TBIT 상태를 변경할 수 없다는 것에 주의 하시오. 만약 소프트웨어적으로 변경하려고 하면, 프로세서는 예측할 수 없는 상태로 진입하게 된다.

인터럽트 디스에이블 bit : I bit와 F bit는 인터럽트 디스에이블 bit이다. 1로 설정될 때, I bit와 F bit는 IRQ와 FIQ 인터럽트를 디스에이블 한다.

모드 bit : M4, M3, M2, M1, M0 bit(M[4:0])는 모드를 나타내는 bit이다. 프로세서의 동작 모드를 결정하며, 표 2-1에 나타나 있다. 32가지의 경우 중 7가지만을 이용한다. M[4:0]에 앞의 7가지 경우를 제

외한 나머지 경우의 값들이 설정되면, 프로세서는 원상복귀가 불가능한 상태로 된다는 것을 주의해야 한다. 이러한 경우에는 리셋을 시켜서 원상복귀를 해야 한다.

예약된 bit : PSR의 나머지 bit들은 예약되어 있다. PSR의 플래그나 컨트롤 bit를 변경할 때는, 사용되지 않는 bit의 값은 그대로 두어야 한다. 또한 프로세서가 1 혹은 0만 읽을 수 있기 때문에, 이 bit들에 특수한 값을 기록 해서도 안된다.

표 2-1. PSR 모드의 bit 값

M[4:0]	Mode	Visible THUMB state registers	Visible ARM state registers
10000	User	R7..R0, LR, SP PC, CPSR	R14..R0, PC, CPSR
10001	FIQ	R7..R0, LR_fiq, SP_fiq PC, CPSR, SPSR_fiq	R7..R0, R14_fiq..R8_fiq, PC, CPSR, SPSR_fiq
10010	IRQ	R7..R0, LR_irq, SP_irq PC, CPSR, SPSR_irq	R12..R0, R14_irq, R13_irq, PC, CPSR, SPSR_irq
10011	Supervisor	R7..R0, LR_svc, SP_svc, PC, CPSR, SPSR_svc	R12..R0, R14_svc, R13_svc, PC, CPSR, SPSR_svc
10111	Abort	R7..R0, LR_abt, SP_abt, PC, CPSR, SPSR_abt	R12..R0, R14_abt, R13_abt, PC, CPSR, SPSR_abt
11011	Undefined	R7..R0 LR_und, SP_und, PC, CPSR, SPSR_und	R12..R0, R14_und, R13_und, PC, CPSR
11111	System	R7..R0, LR, SP PC, CPSR	R14..R0, PC, CPSR

Exceptions

프로그램의 실행이 일시적으로 정지될 때, Exception이 발생하는데, 한 예로, 주변장치로부터 오는 인터럽트를 받을 때이다. exception이 핸들링 되기 전에, 현재의 프로세서 상태는 핸들러 루틴이 끝났을 때 원래의 프로그램이 다시 실행되도록 보관 되어야 한다.

동시에 몇 가지의 exception 발생이 가능하다. 이렇게 되면, 발생된 exception은 우선순위에 따라서 다루어진다. Exception의 우선순위에 대한 내용은 페이지 2-14를 참조하시오.

Exception 모드로의 진입

Exception을 핸들링 할 때, ARM920T의 동작:

1. 링크 레지스터에 다음에 실행될 명령어 주소를 보관한다. Exception이 ARM 상태에서 발생되면, 다음 명령어의 어드레스는 링크 레지스터(즉, 현재의 PC+4 혹은 PC +8 값은 exception에 따라서 결정된다. 표 2-2를 참조하시오.)에 복사된다. Exception이 THUMB 상태에서 발생되면, 링크 레지스터에 기록된 값은 프로그램이 exception에서 복귀한 후에 원래의 장소에서 다시 시작되는 값으로 현재의 PC 옵셋이 된다. 이것은 exception이 어떤 상태에서 발생했는지를 exception 핸들러가 결정할 필요가 없다는 것을 의미한다. 예를 들면, SWI의 경우에, MOVS PC, R14_svc 명령어는 SWI가 ARM 상태에서 발생되었는지, THUMB 상태에서 발생되었는지에 상관없이 항상 다음에 실행될 명령어로 복귀한다.

2. CPSR을 SPSR에 복사한다.
3. CPSR 모드의 bit를 Exception과 관련된 값으로 변경 한다.
4. 관련된 Exception 벡터의 다음 명령어를 PC에 패치 한다.

제어할 수 없는 exception 발생을 막기 위해서 인터럽트 디스에이블 플래그를 설정할 수도 있다.

Exception이 발생할 때 프로세서가 THUMB 상태에 있으면, PC가 exception 벡터 어드레스와 함께 호출될 때 자동적으로 ARM 상태로 변경된다.

Exception 모드에서 탈출

동작이 완료되었을 때, Exception 핸들러의 동작은 아래와 같다:
1. Link 레지스터에서 적정한 옵셋 1개를 뺀 후에, PC로 옮긴다.(옵셋의 값은 exception의 형태에 따라서 변한다.)
2. SPSR을 CPSR로 복사한다.
3. entry가 설정되면, 인터럽트 디스에이블 플래그를 클리어 한다.

주의할 점

SPSR에서 CPSR로 복귀하면서 Exception에 이전의 값을 T bit에 자동적으로 설정하기 때문에, THUMB 상태로의 변환은 필요하지 않다.

Exception 진입/탈출에 대한 요약

표 2-2는 Exception 진입 시에 R14에 보전되는 PC 값과, Exception 핸들러에서 빠져나가는데 필요한 명령어를 나타내고 있다.

	Return Instruction	Previous State		Notes
		ARM R14_x	THUMB R14_x	
BL	MOV PC, R14	PC + 4	PC + 2	1
SWI	MOVS PC, R14_svc	PC + 4	PC + 2	1
UDEF	MOVS PC, R14_und	PC + 4	PC + 2	1
FIQ	SUBS PC, R14_fiq, #4	PC + 4	PC + 4	2
IRQ	SUBS PC, R14_irq, #4	PC + 4	PC + 4	2
PABT	SUBS PC, R14_abt, #4	PC + 4	PC + 4	1
DABT	SUBS PC, R14_abt, #8	PC + 8	PC + 8	3
RESET	NA	–	–	4

표 2-2. Exception 진입/탈출

주의할 점:
1. PC는 prefetch abort를 갖는 BL/SWI/Undefined Instruction fetch의 어드레스이다.
2. PC는 FIQ나 IRQ가 우선순위를 갖기 때문에 수행되지 않았던 명령어의 어드레스이다.
3. PC는 데이터 abort를 발생시키는 호출/저장 명령어의 어드레스이다.
4. 리셋 시에 R14_svc에 저장된 값은 예측할 수 없다.

FIQ

FIQ(빠른 인터럽트 요구) Exception은 데이터 전송, 채널 처리를 위해서 설계 되었으며, ARM 상태에서는 충분한 개별 레지스터를 가진다.

FIQ는 외부의 nFIQ 입력단에 신호 Low가 입력되면 발생한다. 이 입력은 동기 혹은 비동기에 상관없이 ISYNC 입력 신호의 상태에 따라서 결정된다. ISYNC 신호가 Low이면, nFIQ와 nIRQ는 비동기로 간주되며, 인터럽트가 프로세서의 동작에 영향을 미치기 전에, 동기화를 위해서 1 사이클의 지연을 발생한다.

Exception이 ARM 상태에서 발생되는지, THUMB 상태에서 발생되는지에 상관없이, FIQ 핸들러는 아래의 명령어를 수행해서 인터럽트에서 벗어나야 한다.

```
SUBS PC, R14_fiq, #4
```

FIQ는 CPSR의 F 플래그를 셋팅(사용자 모드에서는 불가능함.)해서 디스에이블 한다. 만약 F 플래그가 클리어 되면, ARM920T는 각 명령어의 끝에 FIQ 동기 장치의 출력이 Low인지를 체크한다.

IRQ

IRQ(인터럽트 요청) exception은 nIRQ의 입력 단에 Low 레벨이 들어오면 발생되는 일반적인 인터럽트이다. IRQ는 FIQ 보다 우선순위가 낮으며, FIQ 시퀀스로 진입하면 마스크 된다. 이러한 상황은 오직 특권 모드(비-사용자 모드)에서만 실행이 가능하지만, CPSR의 I bit 설정으로 언제든지 디스에이블이 가능하다.

exception이 ARM 상태 혹은 THUMB 상태에서 발생되는지에 상관없이, IRQ 핸들러는 아래의 명령어를 실행해서, 인터럽트에서 빠져나올 수 있다.

```
SUBS  PC, R14_irq,  #4
```

Abort

Abort는 현재의 메모리 액세스가 완료될 수 없음을 나타낸다. 외부의 ABORT 입력 신호로 발생된다. ARM920T는 메모리 액세스 사이클 동안에 abort exception을 체크한다.

아래와 같이 2가지 형태의 abort가 있다:

☞ Prefetch abort: 명령어 prefetch 동안에 발생

☞ 데이터 abort: 데이터 액세스 동안에 발생

prefetch abort가 발생하면, prefetch 명령어는 유효하다고 표시되지만, exception은 명령어가 파이프라인의 head에 도달할 때까지 실행될 수 없다. 만약 명령어가 실행되지 않으면, abort도 발생하지 않는다.

데이터 abort가 발생하면, 명령어의 형태에 따라서 아래와 같은 상황이 발생한다:

☞ 단일 데이터 전송 명령어(LDR, STR)는 변경된 base 레지스터에 기록 된다: abort 핸들러는 이를 인식할 수 있어야 한다.

☞ 스왑 명령어(SWP)가 실행되지 않더라도 abort가 된다.

☞ 블록 데이터 전송 명령어(LDM, STM)가 완료된다. write-back이 설정되면, base 레지스터는 업데이트 된다. 명령어로 인해서 base 레지스터에 데이터가 덮어쓰기 되더라도, 덮어쓰기가 방지된다. abort가 표시 된 후에 모든 레지스터로의 덮어쓰기가 방지되는데, 이는 R15가 aborted LDM 명령어로 인해서 보전되는 특별한 경우를 의미한다.

abort 메카니즘은 페이징 단위의 가상 메모리 시스템을 적용한다. 이러한 시스템에서, 프로세서는 임의의 어드레스를 발생한다. 어드레스 상의 데이터가 유용하지 않으면, MMU 신호는 abort를 발생한다. abort 핸들러는 abort의 원인을 파악해야 하고, 유용한 데이터를 요구하며, abort 명령어를 재시도 한다. 응용 프로그램은 사용이 가능한 메모리 용량이나, abort에 의해서 영향을 받는 상태에 대해서 알 필요가 없다.

abort에 대한 원인의 수정 후에, 핸들러는 상태(ARM혹은 Thumb)에 상관없이 아래의 명령어를 수행해야 한다.

```
SUBS    PC, R14_abt, #4     ;  prefetch abort
SUBS    PC, R14_abt, #8     ;  데이터 abort
```

위의 명령어는 PC와 CPSR을 원상 복귀하고, abort 된 명령어를 재시도 한다.

소프트웨어 인터럽트

소프트웨어 인터럽트 명령어(SWI)는 supervisor 모드로 진입하는데 사용되며, 일반적으로 특별한 supervisor의 기능을 요청하게 된다. SWI 핸들러는 ARM 혹은 Thumb 상태에 상관없이, 아래의 명령어를 수행해서 복귀해야 한다.

```
MOV     PC, R14_svc
```

이렇게 해서, PC와 CPSR을 원상복귀하고 나서, SWI를 통해서 명령어로 복귀한다.

주의할 점

nFIQ, nIRQ, ISYNC, LOCK, BIGEND, ABORT 핀은 ARM920T CPU 코어에만 있다.

Undefined 명령어

ARM920T가 핸들링이 불가능한 명령어를 받게 되면, undefined 명령어 trap에 빠지게 된다. 이러한 메카니즘은 소프트웨어 에뮬레이션에서 설정된 ARM 혹은 Thumb 명령어를 확장하는데 사용된다.

fail 된 명령어를 에뮬레이션 한 후에, trap 핸들러는 ARM 혹은 Thumb 상태에 상관없이, 아래와 같은 명령어를 수행해야 한다.

```
MOVS    PC, R14_und
```

이러한 명령어는 CPSR을 원상복귀하고, undefined 명령어를 수행하는 명령어로 복귀한다.

Exception 벡터

아래의 표는 exception 벡터 어드레스를 나타내고 있다.

Address	Exception	Mode in Entry
0x00000000	Reset	Supervisor
0x00000004	Undefined instruction	Undefined
0x00000008	Software Interrupt	Supervisor
0x0000000C	Abort (prefetch)	Abort
0x00000010	Abort (data)	Abort
0x00000014	Reserved	Reserved
0x00000018	IRQ	IRQ
0x0000001C	FIQ	FIQ

표 2-3. Exception 벡터

Exception의 우선순위

여러 개의 exception이 동시에 발생하면, 정해진 우선순위 시스템은 핸들링 되는 순서를 결정한다:

가장 우선순위가 높은 exception:
1. 리셋
2. 데이터 abort
3. FIQ
4. IRQ
5. Prefetch abort

가장 우선순위가 낮은 exception
6. Undefined 명령어, 소프트웨어 인터럽트

모든 exception이 동시에 발생할 수는 없다:

Undefined 명령어와 소프트웨어 인터럽트는 각각 현재의 명령어를 특별하게 디코딩하기 때문에 상호 배제된다.

데이터 abort가 FIQ와 동시에 발생하고, FIQ가 인에이블 되어 있으면(즉 CPSR의 F 플래그가 클리어 되면), ARM920T는 데이터 abort 핸들러로 진입하며, FIQ 벡터를 즉시 수행한다. FIQ로부터 복귀하게 되면 데이터 abort 핸들러로 하여금 재 수행을 하게 한다. 데이터 abort가 FIQ보다 높은 우선순위로 놓이게 되면, 전송 에러가 나타나도록 할 필요가 있다. 이러한 exception 진입에 필요한 시간은 최악의 경우에 발생하는 FIQ 지연 계산에 더해져야 한다.

인터럽트 지연

최악의 경우에, FIQ가 인에이블 되어있는 경우에, FIQ의 지연 시간은 동기 장치(비동기이면, Tsyncmax)를 통과하는데 필요한 가장 긴 시간 + 가장 긴 명령어(Tldm, 가장 긴 명령어는 PC를 포함한 모든 레지스터를 호출하는 LDM) 수행 완료 시간 + 데이터 abort 진입(Texc)

에 필요한 시간 + FIQ 진입에 필요한 시간(Tfiq)이다. 이러한 시간이 지나면 ARM920T는 0x1C에서 명령어를 수행한다.

Tsyncmax는 3개의 프로세서 사이클, Tldm은 20개의 사이클, Texc는 3개의 사이클, Tfiq는 2개의 사이클 시간을 갖는다. 그리하여 전체 시간은 28개의 프로세서 사이클 시간이다. 20MHz의 클럭을 연속적으로 이용하면, 1.4us가 약간 넘는다. 최대의 IRQ 지연 시간 계산은 비슷하나, FIQ가 가장 높은 우선순위를 갖는다는 사실과 임의의 시간 동안에 IRQ 핸들링 루틴으로의 진입 지연이 생길 수 있다. FIQ 나 IRQ에 대한 최소의 지연 시간은 synchronizer(Tsyncmin) + Tfiq가 걸리는 최소의 요청 시간으로 구성된다. 이것은 4개의 프로세서 사이클 시간이다.

리셋

nRESET 신호가 Low로 되면, ARM920T는 명령어 수행을 멈추고, 증가되는 word 주소에서 명령어를 패치하게 된다.

nRESET 신호가 High로 되면, ARM920T는 아래와 같은 실행을 하게 된다:

1. PC와 CPSR의 현재 값을 복사해서 R14_svc와 SPSR_svc에 덮어쓰기를 한다. 저장된 PC와 SPSR의 값은 defined 되지 않는다.
2. M[4:0]을 10011(supervisor 모드)로 설정하고, CPSR의 I-bit와 F-bit를 설정하고, CPSR의 T bit를 클리어 한다.
3. 어드레스 0x00에서 다음에 수행할 명령어를 PC로 패치 한다.
4. ARM 상태에서 다시 실행 한다.