ARM, Ltd

- ARM (Acorn RISC Machines // Advanced RISC Machines)

  : ARM사에서 개발한 CPU 디자인의 한 종류

ARM Architecture

넘어감

ARM Architecture

프로그래머를 위한 모델이 아닌, 컴파일러를 위한

나를 이렇게 바라 봐주세요 --> 컴파일러에게 말함

완벽한 추상화에 실패했다고 볼수 있음: 추상화 성공 = 사용자가 아무것도 숙지하지 않고 시켜도 됨

- Programmer's model components : 최소한 이건 알고 나한테 일을 시켜라
  - instruction set: 명령어 집합
  - Data access architecture
  - Operation mode
  - Register architecture
  - Exception handling mechanism

ARM Architecture Comparison

플로팅 포인트 연산: 수치연산, 게이밍등에 사용

가볍게 넘어간 부분도 잘 숙지하시면 좋을 것 같아요. 시험관점에서

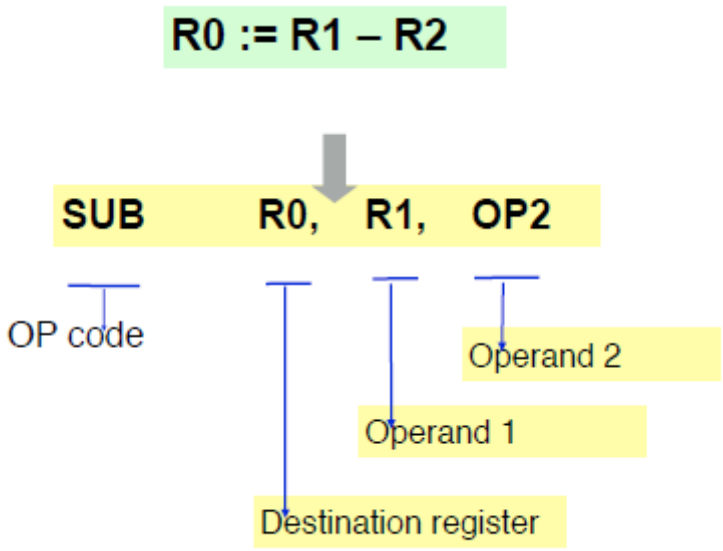
- 표 : 가볍게 보시면 될것같아요

  cache size등이 계속적으로 증가하고 있고, 방대한 구조로 진화하고 있다 정도.

Modern Arm Architecture

넘어감

ARM Instruction Example



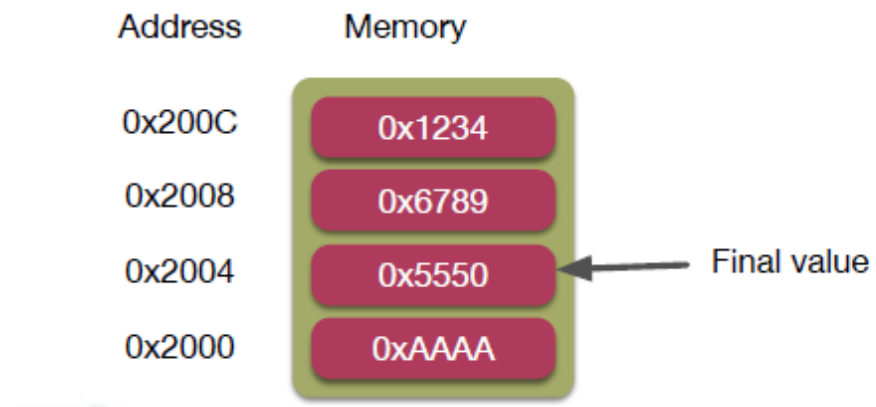
R2는? 이 자리에는 레지스터x address값등을 넣음

메모리 access는 기본적으로 허용되지x but 딱 두개 : LDR / STR

Destination은 사실은 Source가 되겠죠

ARM Assembly Instructions

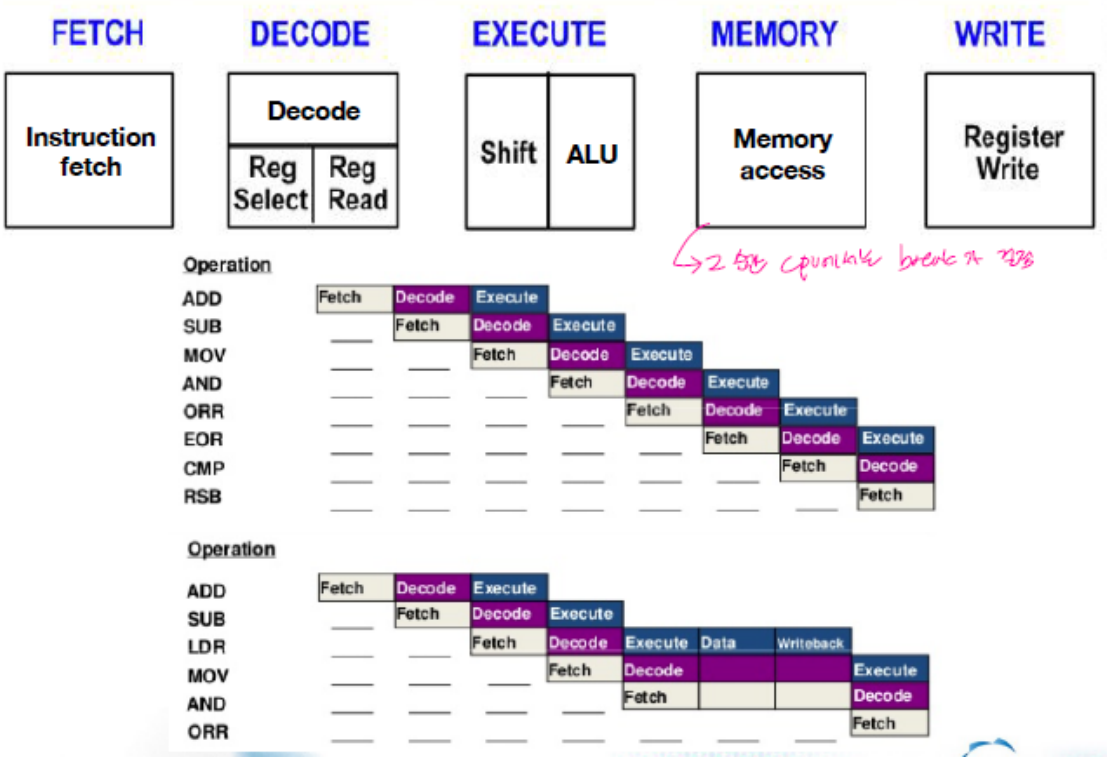
Address	Instruction	Description
0x1000	LDR R0, [R4, R5]	; R4 = 0x2000, R5 = 0xC ; Load the value in [0x200C] to R0
0x1004	LDR R1, [R4, #8]	; Load the value in [0x2008] to R1
0x1008	ADD R2, R0, #5	; R2 := R0 + 5
0x100C	SUB R3, R1, R2	; R3 := R1 – R2
0x1010	STR R3, [R4, #4]	; Store the value in R3 to [0x2004]



ARM Pipelines

5 stage pip lines

- 레지스터와 레지스터 명령어는 실행속도가 매우 빠른 반면, 레지스터와 메모리간의 명령어는 외부 버스를 통해 접근하기 때문에 상대적으로 속도가 느리고, 파이프 라인 단계도 추가됨



ARM Instruction Set

- 32bit 명령어: ARM instruction
- 16bit 명령어: Thumb instruction : 전력소모가 줄어든다

이정도로만 간단하게 넘어갈게요 (읽고 넘김)

32 Bits ARM Instructions Set

ARM Instruction Types

마음만 먹으면 외울수 있어요,, (외우라고는 안함)

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

KESL

ARM / Thumb Interwork

이미 한번 설명드렸죠 가볍게 읽고 넘

- ARM state
  - 32bit ARM 명령어를 수행할 준비
  - 예외 발생시, 현재 상태에 상관 없이 CPU상태가 ARM 상태로 변경됨
- Thumb state
  - 16bit 명령어 집합 사용
  - BX명령으로 run-time 으로 CPU를 변경 (?)

Operating Mode

명령어로 mode가 변경되는 것이 아닌, exception으로 특별한 상황이 발생했을 때 mode가 변경함

-- user mode --

- user mode: user를 신뢰하지 못함(user의 실수, user의 악의)

-- privileged mode --

아래 두개는

- FIQ (Fast Interrupt reQuest) : 센서 같은 io 장치 쓸때, 준비 완료, 급행모드
- IRQ (Interrupt ReQuest): io 장치
- SVC (Supervisor): 소프트웨어 인터럽트 (커널에서 사용하는 명령어)
- (ABT)Abort Mode: os의 메모리 protection(허가받지 않은 access에 대해서)
- (UND)Undefined Mode: fetch해서 decode 했더니 내가 모르는 명령어야!
- (SYS)System Mode: svc(privileged mode 최초 진입로(소프트웨어적으로만, io장치 인터럽트는 아님)) -> system mode

mode 마다 우선순위가 있음 -> 숫자 낮은거 우선순위

Operating Mode Switch

- exception

  예외 발생시, HW는 자동으로 작동모드를 적절하게 변환

  예외 유형이 작동 모드와 밀접하게 일치함

- system call

  여러가지 system call을 조합해서 library함수도 만듦

  system call에 들어가보면 SWI (previleged mode)

Exception Vectors

- interrupt service routine
- Prefetch Abort: instruction 을 읽어올 때

4byte 안에 exception handler를

ARM Registers

- 범용 레지스터

  30개 레지스터

- 특수 레지스터

  • Program Counter (PC) : R15 • Current Program Status Register (CPSR) • Saved Program Status Register (SPSR)

  - 모드 스위치가 발생했을 때 직전 동작 모드의 CPSR 저장

ARM Registers for Operating Modes

Thumb Mode Registers

Stack Pointer(R13)

- 특수 목적 전용 범용 레지스터

  •현재 SP 위치 저장 •각 작동 모드 자체 SP 레지스터 보유 •ARM은 PUSH/POP 지침을 제공하지 않음 • LDM/STM 지침을 사용하여 스택 작동 구현

Link Register (LR or R14)

- 함수 호출 또는 예외가 발생했을 시 복귀 주소를 저장하는 레지스터
- BL 명령어로 return 주소를 LR에 저장

  bl: 명령어 뒤에 지정된 상수값에 해당하는 주소로 분기하되, 현재의 PC값 + 2번지의 복귀 주소값을 가지는 링크 레지스터에 남겨놓는 방식

- LR에 저장된 데이터를 PC로 가져와서 서브루틴에서 복귀
- LR에 저장된 주소는 데이터 처리 지침에 따라 수정할 수 있음

- 각 작동 모드 자체 LR레지스터 보유

Program Counter (PC or R15)

- 주행해야할 주소를 가지고 있는 레지스터
- PC는 명령어를 사용하여 직접 수정할 수 있음
- 시스템 내 PC레지스터 한개만

Program Status Register

- PSR
    - CPSR 1개
    - SPSR 5개 (각 작동 모드에)
- PSR 레지스터 필드
    - Condition flag
        - ALU 처리 결과 flag
    - Control bits
        - CPU 상태 제어