ARM architecture

노트북:	SW		
만든 날짜:	2018-04-30 오후 3:14	**수정한 날짜:**	2018-04-30 오후 8:42
작성자:	정상요		

개념(cpsr 이 중요) + 실습(예제 7개 -> cpsr 확인)

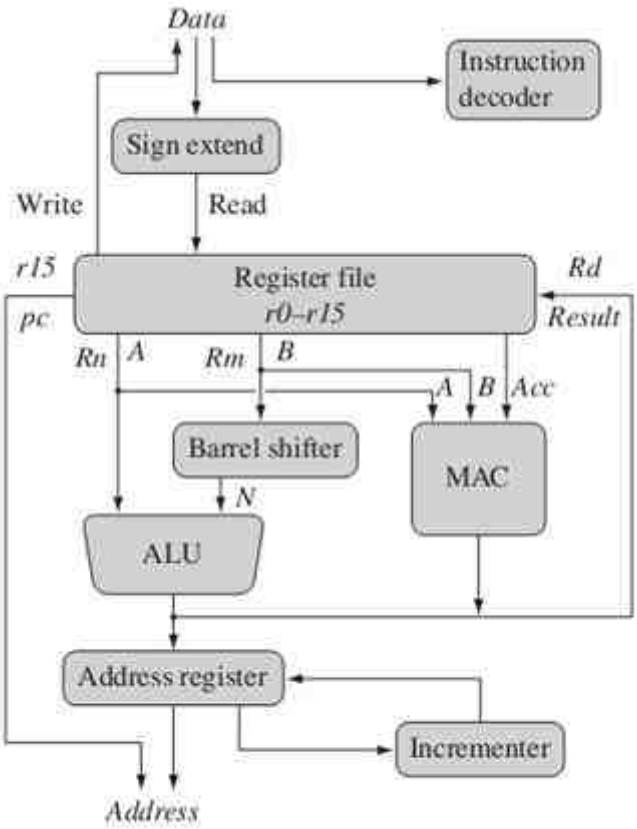
어셈블리어 해석이 목표!!! -> 어셈블리어는 속도측면에서 효율적이다.
MCU : 어셈블리어를 해석할 줄 알아야 한다.
DSP, FPGA : 어셈블리어를 직접 작성할 줄 알아야한다.

- Embedded System 이란???

모든 전자기기

- ARM 전체적인 구조

Data bus, Address bus, Register file, MAC



Simplified von-newman representation of ARM core

MAC의 중요성

MAC의 유무에 따라 ARM, DSP로 나누어 진다.(MAC 존재 : DSP, MAC 존재X : ARM)
DSP : 고속연산하는데 있어 적합

ARM : General Register(r0 ~ r15 + cpsr)

CPSR Register

Modern Status Register

비교연산

-> ASM
if & lt, gt, eq, ne, le, ge 와 함께 사용
lt : less than
gt : greater than
eq : equal to
ne : not equal to
le : less than or equal to
ge : greater than or equal to

1. add r0, r1, r2 -> r0 = r1 + r2
2. subgt r3, r3, #1 -> r3 = r3 - 1
3. rsble r4, r5, #5 -> r4 = 5 -r5
4. and r0, r1, r2 -> r0 = r1, r2 AND 연산
5. biceq r2, r3, #7 ->

ARM 환경설정(띄어쓰기 주의)

1. sudo apt-get update
   sudo apt-get install qemu-user-static qemu-system
   sudo apt-get install gcc-arm-linux-gnueabi
2. C 파일 작성
3. arm-linux-gnueabi-gcc -g 소스파일
4. sudo apt-get install gdb-multiarch
5. 터미널을 2개 띄운다.
6. A 터미널에서 아래 명령어를 수행한다.
   qemu-arm-static -g 1234 -L /usr/arm-linux-gnueabi ./a.out --- 1234 : port number
7. B 터미널에서 아래 명령어를 수행한다.
8. gdb-multiarch
9. file a.out
10. target remote localhost:1234
11. b main
12. c

이후부터 디버깅을 진행하면 된다.
다른 모든 프로그램도 이러한 방법으로 디버깅을 수행할 수 있다.
이제 ARM Architecture에 대해 Deep Down 할 수 있는 기점이 마련되었다.

gdb 명령어 : info reg(info register), disas, l, ni, si

ARM 은 무조건 4byte 단위로 움직인다.
intel은 1byte로 움직일 수 있다.

ARM사가 하는 일 : 칩설계 -> FPGA로 CPU로직을 설계 : FPGA로 논리로 TR 로직상 10억개 배치(손으로 직접 10억개하면)RTL 설계했다고 한다.
팹리스에 넘겨준다. 팹리스는 해석하는 툴에 RTL을 넣으면 반도체직접함

결국 ARM사는 FPGA 한다.

CHIP에 동작해야하는 프로그램이 설계가 다 되어있음
FPGA : 반도체 직접 시켜주는 일

ARM사 : 설계업체(설계도만 만듬)

TI : 칩자체를 양산해 낸다. -> 패리패럴을 붙힌다. CPU 여러개를 붙힌다. (DSP, CPU
Fabless :

cmp, tst, teq, cmn