TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 최대성
c3d4s19@naver.com

2018.04.30. – 045일차 수업

ARM 어셈블리어 수업

ARM은 함수의 인자를 Register로 전달

Intel은 함수의 인자를 Stack으로 전달

push {r11, lr}

r11 -> bp / lr -> 복귀주소 저장

어셈에서 ; 뒤에 내용은 주석임

인자가 4개를 넘어가면 Register 대신 Stack을 사용함

함수 인자는 4개 이하로 정하는게 속도가 빠름

넘어가면 구조체 인자 이용 하면 해결

함수의 리턴값은 r0에 세팅된다

어셈 코딩할때 r0, r1, r2, r3 사용 피하는게 좋음

되도록 r4, r5, r6, r7, r8, r9 사용

r10, r11, r12는 다른 용도가 있긴 하지만 중요한게 아니므로 그냥 사용해도 된다

그런데 r7은 System Call 에서 사용하므로 주의

gdb 명령어 정리 링크

http://mintnlatte.tistory.com/581

명령어

mov r0, #0xff, 8

-> 0xff를 8bit를 버리지 않고 순환시켜서 >> shift함 0x ff00 0000

mul r0, r1, r2

-> r0 = r1 * r2

mla r0, r1, r2, r3

-> r0 = r1 * r2 + r3

umull r0, r1, r2, r3

-> (r0 + (r1<<8)) == r2 * r3

r0 = (하위 16byte), r1 = (상위 16byte)

umlal r0, r1, r2, r3

-> (r0 + (r1<<8)) += (r2 * r3)

r0 = r0 + (하위 16byte), r1 = r1 + (상위 16byte)

add r0, r1, r2, lsl r3

-> r0 = r1 + r2<<r3

add r0, r1, r2, asr r3

-> r0 = r1 + r2>>r3

mrs r0, cpsr

-> r0 = cpsr

str r0, [r1]

-> *r1 = r0

ldr r0, [r1]

-> r0 = *r1

ldr r0, [r1,r2]

-> r0 = *(r1 + r2)

[r1,r2]!

-> r1 += r2 상태로 고정

(!가 붙으면 연산한 위치로 옮김)

ldr r0, [r1], r2

-> r0 = *r1, r1 += r2 / r0에 *r1값을 넣고 r1의 주소값을 r2만큼 증가

접미사 b -> byte단위로

stmia r0, {r1, r2, r3}

-> r0 주소 구조체 맴버들을 r1, r2, r3로 초기화

ldmia r0, {r1, r2, r3}

-> r1, r2, r3 레지스터를 r0 주소 구조체 맴버들로 초기화