


Xilinx Zynq FPGA,TI DSP, MCU 기반의 프로그래밍 전문가 과정

날 짜 : 2018 . 4. 30

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 정한별
hanbulkr@gmail.com

< Arm architecture>

임베디드 → 각종 전자기기 다. 임베디드가 기술이 들어갔다고생각 하면 됨. (HW +SW)

[ARM 환경설정]
sudo apt-get update
sudo apt-get install qemu-user-static qemu-system
sudo apt-get install gcc-arm-linux-gnueabi
sudo apt-get install gdb-multiarch
(이후에 아무런 C 소스 파일을 작성한다.)

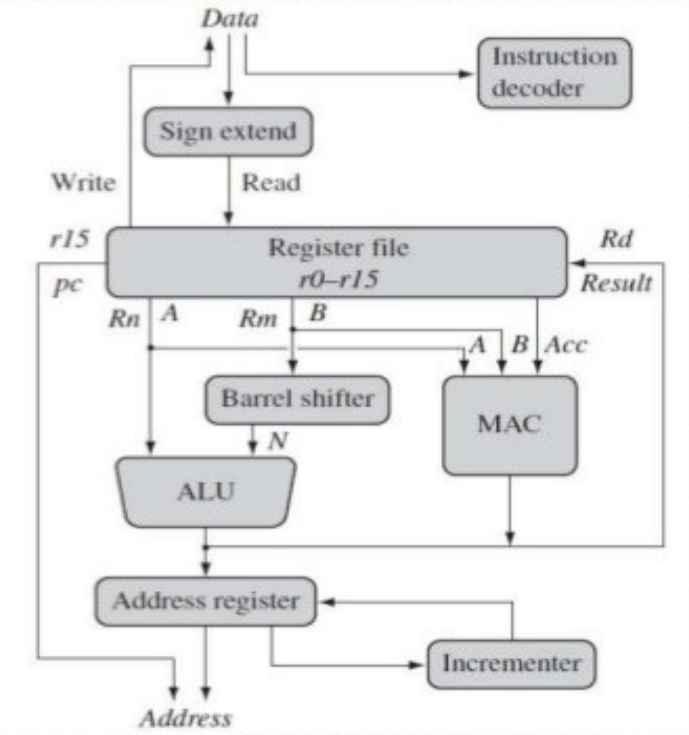
컴파일 하기 : arm-linux-gnueabi-gcc -g 소스파일
실행 하기 : qemu-arm-static -L /usr/arm-linux-gnueabi ./a.out

(터미널을 2 개 띄운다)
A 터미널에서 아래 명령어를 수행한다.
qemu-arm-static -g 1234 -L /usr/arm-linux-gnueabi ./a.out
(1234 는 디버깅 로컬 포트를 열어서 다른곳에서 접근 가능하게 한다.)

B 터미널에서 아래 명령어를 수행한다.
gdb-multiarch
file a.out
target remote localhost:1234 // 로컬포트로 접속해 디버깅을 실시간으로 건드려봄.
b main
c

(이후 info reg, disas, ni,si 등을 이용하면서 디버깅을 한다.)

ARM core Data flow Model: A programmer can think of an ARM core as functional units connected by data buses, as shown in Figure.



ARM core dataflow model

< ARM 코어의 데이타 흐름도 >

– MAC 곱셈기(옵션) → MAC 이 있으면 DSP 라고 생각하면 됨.

→ 연산하는 clock 횟수를 줄이려고(모든 곱셈+덧셈을 병렬로, 1clock 으로 끝남)

→ 병렬은 4 개(simd)

→ 제일 좋은 구간은? sin(x)^t * e^iyt

테일러급수 * 오일러

cpu(f, 동작횟수)

샘플링타임→ 이용해서 아날로그를 계산가능해짐. (작으면 오차율이 줄어듬)

[General Register]

r0
r1
r2
r3
r4
r5
r6
r7
r8
r9
r10
r11
r12
r13 sp
r14 lr
r15 pc

cpsr
-

Figure 2.2 Registers available in *user* mode.

– 16 개의 data registers[r0 – r15] 와 2 개의 status registers 로 구성.
– 한번에 18 개까지 활성화 가능

>> 특수 목적을 갖는 레지스터

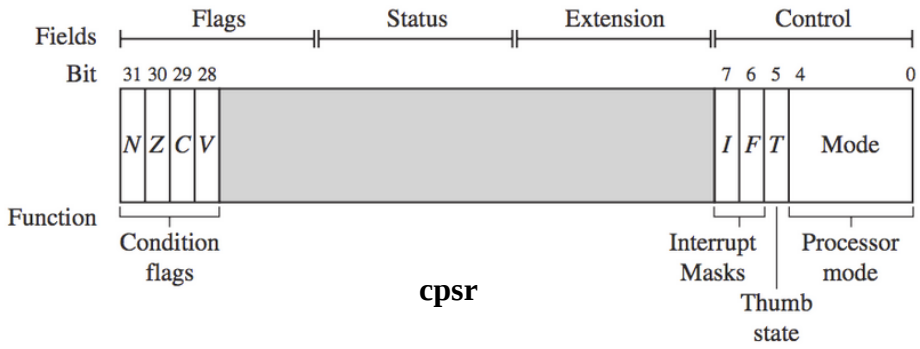
– r13: SP(stack pointer). 현재 모든 프로세서 모드의 스택 위 주소값 저장
– r14: LR(Link Register). 코어가 서브루틴을 호출 할 때마다 그 복귀주소를 저장.
– r15: PC(Program Counter). 프로세서가 읽어들인 다음 명령어의 주소를 저장.

** r13, r14 는 범용 레지스터로도 사용하며 프로세서 모드가 바뀌면 뱅크에 저장되므로 특히나 유용하게 사용할 수 있음. but, r13 은 OS 에서 유효한 스택 프레임을 가리키고 있는 것으로 가정하여 동작하므로 위험함.

>> 상태레지스터

– cpsr: Current Program Status Register. **현재의 프로그램 상태 레지스터.(32bit register)**
- Register file 안에 위치.
- 8bit 씩, 4 개의 영역으로 나뉨.
(flag(8bit), Status(8bit), Extension(8bit), Control(8bit)).

– spsr: Stored Program Status Register. **이전에 저장된 프로그램 상태 레지스터.**



cpsr

Flag	Flag name	Set when
Q	Saturation	the result causes an overflow and/or saturation
V	oVerflow	the result causes a signed overflow
C	Carry	the result causes an unsigned carry
Z	Zero	the result is zero, frequently used to indicate equality
N	Negative	bit 31 of the result is a binary 1

- **nzCvq →** C 만 대문자인 이유는 플래그 값이 1 임. 플래그 값이 1 일 때는 대문자로 표현.
  ***인터럽트의 경우, 대문자는 인터럽트 비활성을 의미.**
- **SVC :** Mode[4:0] 이 supervior 모드임 이다.
- **명령어 실행 전** core 는 자신이 가지고 있는 조건인자와 cpsr 의 상태플래그를 **비교** 후 일치하면 실행.
- **조건인자가 없다면** "AL" 으로 설정되어 항상 실행된다.

Table 2.5 Condition mnemonics.

Mnemonic	Name	Condition flags
EQ	equal	*Z*
NE	not equal	*z*
CS HS	carry set/unsigned higher or same	*C*
CC LO	carry clear/unsigned lower	*c*
MI	minus/negative	*N*
PL	plus/positive or zero	*n*
VS	overflow	*V*
VC	no overflow	*v*
HI	unsigned higher	*zC*
LS	unsigned lower or same	*Z* or *c*
GE	signed greater than or equal	*NV* or *nv*
LT	signed less than	*Nv* or *nV*
GT	signed greater than	*NzV* or *nzv*
LE	signed less than or equal	*Z* or *Nv* or *nV*
AL	always (unconditional)	ignored

can see from the figure the processor is in *supervisor* (*SVC*) mode since the mode[4:0] is equal to binary 10011.

ARM 아키텍쳐 명령어.

Table 3.1 ARM instruction set.

Mnemonics	ARM ISA	Description
ADC	v1	add two 32-bit values and carry
ADD	v1	add two 32-bit values
AND	v1	logical bitwise AND of two 32-bit values
B	v1	branch relative +/− 32 MB
BIC	v1	logical bit clear (AND NOT) of two 32-bit values
BKPT	v5	breakpoint instructions
BL	v1	relative branch with link
BLX	v5	branch with link and exchange
BX	v4T	branch with exchange
CDP CDP2	v2 v5	coprocessor data processing operation
CLZ	v5	count leading zeros
CMN	v1	compare negative two 32-bit values
CMP	v1	compare two 32-bit values
EOR	v1	logical exclusive OR of two 32-bit values
LDC LDC2	v2 v5	load to coprocessor single or multiple 32-bit values
LDM	v1	load multiple 32-bit words from memory to ARM registers
LDR	v1 v4 v5E	load a single value from a virtual address in memory
MCR MCR2 MCRR	v2 v5 v5E	move to coprocessor from an ARM register or registers
MLA	v2	multiply and accumulate 32-bit values
MOV	v1	move a 32-bit value into a register
MRC MRC2 MRRC	v2 v5 v5E	move to ARM register or registers from a coprocessor
MRS	v3	move to ARM register from a status register (*cpsr* or *spsr*)
MSR	v3	move to a status register (*cpsr* or *spsr*) from an ARM register
MUL	v2	multiply two 32-bit values
MVN	v1	move the logical NOT of 32-bit value into a register
ORR	v1	logical bitwise OR of two 32-bit values
PLD	v5E	preload hint instruction
QADD	v5E	signed saturated 32-bit add
QDADD	v5E	signed saturated double and 32-bit add
QDSUB	v5E	signed saturated double and 32-bit subtract
QSUB	v5E	signed saturated 32-bit subtract
RSB	v1	reverse subtract of two 32-bit values
RSC	v1	reverse subtract with carry of two 32-bit integers
SBC	v1	subtract with carry of two 32-bit values
SMLA*xy*	v5E	signed multiply accumulate instructions ((16 × 16) + 32 = 32-bit)
SMLAL	v3M	signed multiply accumulate long ((32 × 32) + 64 = 64-bit)
SMLAL*xy*	v5E	signed multiply accumulate long ((16 × 16) + 64 = 64-bit)
SMLAW*y*	v5E	signed multiply accumulate instruction (((32 × 16) ≫ 16) + 32 = 32-bit)
SMULL	v3M	signed multiply long (32 × 32 = 64-bit)
SMUL*xy*	v5E	signed multiply instructions (16 × 16 = 32-bit)
SMULW*y*	v5E	signed multiply instruction ((32 × 16) ≫ 16 = 32-bit)
STC STC2	v2 v5	store to memory single or multiple 32-bit values from coprocessor
STM	v1	store multiple 32-bit registers to memory
STR	v1 v4 v5E	store register to a virtual address in memory
SUB	v1	subtract two 32-bit values
SWI	v1	software interrupt
SWP	v2a	swap a word/byte in memory with a register, without interruption
TEQ	v1	test for equality of two 32-bit values
TST	v1	test for bits in a 32-bit value
UMLAL	v3M	unsigned multiply accumulate long ((32 × 32) + 64 = 64-bit)
UMULL	v3M	unsigned multiply long (32 × 32 = 64-bit)

<add>

```
jhb@onestar:~/my/Homework/hanbyuljung/class/class_45_me$ qemu-arm-static -L /usr/arm-linu
x-gnueabi ./a.out
r0 = 114
```

```
Dump of assembler code for function main:
   0x00010438 <+0>:     push    {r11, lr}
   0x0001043c <+4>:     add     r11, sp, #4
=> 0x00010440 <+8>:     mov     r1, #77 ; 0x4d
   0x00010444 <+12>:    mov     r2, #37 ; 0x25
   0x00010448 <+16>:    add     r0, r1, r2
   0x0001044c <+20>:    mov     r3, r0
   0x00010450 <+24>:    mov     r1, r3
   0x00010454 <+28>:    ldr     r0, [pc, #12]   ; 0x10
   0x00010458 <+32>:    bl      0x102e0 <printf@plt>
   0x0001045c <+36>:    mov     r3, #0
   0x00010460 <+40>:    mov     r0, r3
   0x00010464 <+44>:    pop     {r11, pc}
   0x00010468 <+48>:    ldrdeq  r0, [r1], -r12
End of assembler dump.
(gdb) info reg
r0             0x1      1
r1             0xf6ffef64       -150999196
r2             0xf6ffef6c       -150999188
r3             0x10438  66616
r4             0x1046c  66668
r5             0x0      0
r6             0x10310  66320
r7             0x0      0
r8             0x0      0
r9             0x0      0
r10            0xf67fe000       -159391744
r11            0xf6ffee14       -150999532
r12            0xf6ffee90       -150999408
sp             0xf6ffee10       0xf6ffee10
lr             0xf6688d14       -160920300
pc             0x10440  0x10440 <main+8>
cpsr           0x60000010       1610612752
```

```
r0             0x1      1
r1             0x4d     77
r2             0x25     37
```

```
r0             0x72     114
r1             0x4d     77
r2             0x25     37
r3             0x10438  66616
```

<subgt>

```
jhb@onestar:~/my/Homework/hanbyuljung/class/class_45_me$ qemu-arm-static -L /usr/arm-linu
x-gnueabi ./a.out
r3 = 33
```

```
(gdb) info reg
r0             0x1      1
r1             0x4d     77
r2             0x25     37
r3             0x22     34
r4             0x10474  66676
r5             0x0      0
r6             0x10310  66320
r7             0x0      0
r8             0x0      0
r9             0x0      0
r10            0xf67fe000       -159391744
r11            0xf6ffee14       -150999532
r12            0xf6ffee90       -150999408
sp             0xf6ffee10       0xf6ffee10
lr             0xf6688d14       -160920300
pc             0x1044c  0x1044c <main+20>
cpsr           0x60000010       1610612752
(gdb) disas
Dump of assembler code for function main:
   0x00010438 <+0>:     push    {r11, lr}
   0x0001043c <+4>:     add     r11, sp, #4
   0x00010440 <+8>:     mov     r1, #77 ; 0x4d
   0x00010444 <+12>:    mov     r2, #37 ; 0x25
   0x00010448 <+16>:    mov     r3, #34 ; 0x22
=> 0x0001044c <+20>:    cmp     r1, r2
   0x00010450 <+24>:    bls     0x10458 <main+32>
   0x00010454 <+28>:    subgt   r3, r3, #1
   0x00010458 <+32>:    mov     r1, r3
   0x0001045c <+36>:    ldr     r0, [pc, #12]   ; 0x10470 <main+
   0x00010460 <+40>:    bl      0x102e0 <printf@plt>
   0x00010464 <+44>:    mov     r3, #0
   0x00010468 <+48>:    mov     r0, r3
   0x0001046c <+52>:    pop     {r11, pc}
   0x00010470 <+56>:    andeq   r0, r1, r4, ror #9
```

```
(gdb) disas
Dump of assembler code for function main:
   0x00010438 <+0>:     push    {r11, lr}
   0x0001043c <+4>:     add     r11, sp, #4
   0x00010440 <+8>:     mov     r1, #77 ; 0x4d
   0x00010444 <+12>:    mov     r2, #37 ; 0x25
   0x00010448 <+16>:    mov     r3, #34 ; 0x22
   0x0001044c <+20>:    cmp     r1, r2
   0x00010450 <+24>:    bls     0x10458 <main+32>
   0x00010454 <+28>:    subgt   r3, r3, #1
=> 0x00010458 <+32>:    mov     r1, r3
   0x0001045c <+36>:    ldr     r0, [pc, #12]   ; 0
   0x00010460 <+40>:    bl      0x102e0 <printf@pl
   0x00010464 <+44>:    mov     r3, #0
   0x00010468 <+48>:    mov     r0, r3
   0x0001046c <+52>:    pop     {r11, pc}
   0x00010470 <+56>:    andeq   r0, r1, r4, ror #9
End of assembler dump.
(gdb) info reg
r0             0x1      1
r1             0x4d     77
r2             0x25     37
r3             0x21     33
r4             0x10474  66676
r5             0x0      0
r6             0x10310  66320
r7             0x0      0
r8             0x0      0
r9             0x0      0
r10            0xf67fe000       -159391744
r11            0xf6ffee14       -150999532
r12            0xf6ffee90       -150999408
sp             0xf6ffee10       0xf6ffee10
lr             0xf6688d14       -160920300
pc             0x10458  0x10458 <main+32>
cpsr           0x20000010       536870928
```

subgt : gt→ greater than sub 하란 의미.

<rsble> - reverse subtract(not sub)

```
jhb@onestar:~/my/Homework/hanbyuljung/class/class_45_me$ qemu-arm-static -L /usr/arm-linu
x-gnueabi ./a.out
r4 = 2
```

```
(gdb) disas
Dump of assembler code for function main:
   0x00010438 <+0>:     push    {r4, r5, r11, lr}
   0x0001043c <+4>:     add     r11, sp, #12
   0x00010440 <+8>:     mov     r1, #77 ; 0x4d
   0x00010444 <+12>:    mov     r2, #37 ; 0x25
   0x00010448 <+16>:    mov     r3, #34 ; 0x22
   0x0001044c <+20>:    mov     r5, #3
   0x00010450 <+24>:    mov     r3, r1
=> 0x00010454 <+28>:    cmp     r2, r3
   0x00010458 <+32>:    bhi     0x10460 <main+40>
   0x0001045c <+36>:    rsble   r4, r5, #5
   0x00010460 <+40>:    mov     r3, r4
   0x00010464 <+44>:    mov     r1, r3
   0x00010468 <+48>:    ldr     r0, [pc, #12]   ; 0x1047
   0x0001046c <+52>:    bl      0x102e0 <printf@plt>
   0x00010470 <+56>:    mov     r3, #0
   0x00010474 <+60>:    mov     r0, r3
   0x00010478 <+64>:    pop     {r4, r5, r11, pc}
   0x0001047c <+68>:    strdeq  r0, [r1], -r0   ; <UNPRE
End of assembler dump.
(gdb) info reg
r0             0x1      1
r1             0x4d     77
r2             0x25     37
r3             0x4d     77
r4             0x10480  66688
r5             0x3      3
r6             0x10310  66320
r7             0x0      0
r8             0x0      0
r9             0x0      0
r10            0xf67fe000       -159391744
r11            0xf6ffee14       -150999532
r12            0xf6ffee90       -150999408
sp             0xf6ffee08       0xf6ffee08
lr             0xf6688d14       -160920300
pc             0x10454  0x10454 <main+28>
cpsr           0x60000010       1610612752
```

```
(gdb) disas
Dump of assembler code for function main:
   0x00010438 <+0>:     push    {r4, r5, r11, lr}
   0x0001043c <+4>:     add     r11, sp, #12
   0x00010440 <+8>:     mov     r1, #77 ; 0x4d
   0x00010444 <+12>:    mov     r2, #37 ; 0x25
   0x00010448 <+16>:    mov     r3, #34 ; 0x22
   0x0001044c <+20>:    mov     r5, #3
   0x00010450 <+24>:    mov     r3, r1
   0x00010454 <+28>:    cmp     r2, r3
   0x00010458 <+32>:    bhi     0x10460 <main+40>
   0x0001045c <+36>:    rsble   r4, r5, #5
=> 0x00010460 <+40>:    mov     r3, r4
   0x00010464 <+44>:    mov     r1, r3
   0x00010468 <+48>:    ldr     r0, [pc, #12]   ; 0x1047c <main+6
   0x0001046c <+52>:    bl      0x102e0 <printf@plt>
   0x00010470 <+56>:    mov     r3, #0
   0x00010474 <+60>:    mov     r0, r3
   0x00010478 <+64>:    pop     {r4, r5, r11, pc}
   0x0001047c <+68>:    strdeq  r0, [r1], -r0   ; <UNPREDICTABLE>
End of assembler dump.
(gdb) info reg
r0             0x1      1
r1             0x4d     77
r2             0x25     37
r3             0x4d     77
r4             0x2      2
r5             0x3      3
r6             0x10310  66320
r7             0x0      0
r8             0x0      0
r9             0x0      0
r10            0xf67fe000       -159391744
r11            0xf6ffee14       -150999532
r12            0xf6ffee90       -150999408
sp             0xf6ffee08       0xf6ffee08
lr             0xf6688d14       -160920300
pc             0x10460  0x10460 <main+40>
cpsr           0x80000010       -2147483632
```

<biceq> - bitclear equal(42&~(2^3-1))8 의 배수로 짤림그래서 40 에서 짜름.

```
jhb@onestar:~/my/Homework/hanbyuljung/class/class_45_me$ qemu-arm-static -L /usr/arm-linu
x-gnueabi ./a.out
00000000000000000000000000101000
```

```
Dump of assembler code for function main:
   0x000104cc <+0>:     push    {r11, lr}
   0x000104d0 <+4>:     add     r11, sp, #4
   0x000104d4 <+8>:     mov     r0, #7
   0x000104d8 <+12>:    mov     r1, #7
   0x000104dc <+16>:    mov     r3, r1
=> 0x000104e0 <+20>:    cmp     r0, r3
   0x000104e4 <+24>:    bne     0x104f0 <main+3
   0x000104e8 <+28>:    mov     r3, #42 ; 0x2a
   0x000104ec <+32>:    biceq   r2, r3, #7
   0x000104f0 <+36>:    mov     r3, r2
   0x000104f4 <+40>:    mov     r0, r3
   0x000104f8 <+44>:    bl      0x10468 <show_r
   0x000104fc <+48>:    mov     r3, #0
   0x00010500 <+52>:    mov     r0, r3
   0x00010504 <+56>:    pop     {r11, pc}
End of assembler dump.
(gdb) info reg
r0             0x7      7
r1             0x7      7
r2             0xf6ffef6c       -150999188
r3             0x7      7
r4             0x10508  66824
r5             0x0      0
r6             0x10340  66368
r7             0x0      0
r8             0x0      0
r9             0x0      0
r10            0xf67fe000       -159391744
r11            0xf6ffee14       -150999532
r12            0xf6ffee90       -150999408
sp             0xf6ffee10       0xf6ffee10
lr             0xf6688d14       -160920300
pc             0x104e0  0x104e0 <main+20>
cpsr           0x60000010       1610612752
```

```
Dump of assembler code for function main:
   0x000104cc <+0>:     push    {r11, lr}
   0x000104d0 <+4>:     add     r11, sp, #4
   0x000104d4 <+8>:     mov     r0, #7
   0x000104d8 <+12>:    mov     r1, #7
   0x000104dc <+16>:    mov     r3, r1
   0x000104e0 <+20>:    cmp     r0, r3
   0x000104e4 <+24>:    bne     0x104f0 <main+3
   0x000104e8 <+28>:    mov     r3, #42 ; 0x2a
=> 0x000104ec <+32>:    biceq   r2, r3, #7
   0x000104f0 <+36>:    mov     r3, r2
   0x000104f4 <+40>:    mov     r0, r3
   0x000104f8 <+44>:    bl      0x10468 <show_r
   0x000104fc <+48>:    mov     r3, #0
   0x00010500 <+52>:    mov     r0, r3
   0x00010504 <+56>:    pop     {r11, pc}
End of assembler dump.
(gdb) ni
0x000104f0 in main ()
(gdb) info reg
r0             0x7      7
r1             0x7      7
r2             0x28     40
r3             0x2a     42
r4             0x10508  66824
r5             0x0      0
r6             0x10340  66368
r7             0x0      0
r8             0x0      0
r9             0x0      0
r10            0xf67fe000       -159391744
r11            0xf6ffee14       -150999532
r12            0xf6ffee90       -150999408
sp             0xf6ffee10       0xf6ffee10
lr             0xf6688d14       -160920300
pc             0x104f0  0x104f0 <main+36>
cpsr           0x60000010       1610612752
```

< orr > -or

```
jhb@onestar:~/my/Homework/hanbyuljung/class/class_45_me$ qemu-arm-static -L /usr/arm-linu
x-gnueabi ./a.out
00000000000000000000000000101111
```

```
Dump of assembler code for function main:
   0x000104cc <+0>:     push    {r4, r5, r11, lr}
   0x000104d0 <+4>:     add     r11, sp, #12
=> 0x000104d4 <+8>:     mov     r5, #3
   0x000104d8 <+12>:    mov     r3, r1
   0x000104dc <+16>:    cmp     r0, r3
   0x000104e0 <+20>:    bne     0x104ec <main+32>
   0x000104e4 <+24>:    mov     r3, #44 ; 0x2c
   0x000104e8 <+28>:    orr     r2, r3, r5
   0x000104ec <+32>:    mov     r3, r2
   0x000104f0 <+36>:    mov     r0, r3
   0x000104f4 <+40>:    bl      0x10468 <show_reg>
   0x000104f8 <+44>:    mov     r3, #0
   0x000104fc <+48>:    mov     r0, r3
   0x00010500 <+52>:    pop     {r4, r5, r11, pc}
End of assembler dump.
(gdb) info reg
r0             0x1      1
r1             0xf6ffef64       -150999196
r2             0xf6ffef6c       -150999188
r3             0x104cc  66764
r4             0x10504  66820
r5             0x0      0
r6             0x10340  66368
r7             0x0      0
r8             0x0      0
r9             0x0      0
r10            0xf67fe000       -159391744
r11            0xf6ffee14       -150999532
r12            0xf6ffee90       -150999408
sp             0xf6ffee08       0xf6ffee08
lr             0xf6688d14       -160920300
pc             0x104d4  0x104d4 <main+8>
cpsr           0x60000010       1610612752
```

```
Dump of assembler code for function main:
   0x000104cc <+0>:     push    {r4, r5, r11, lr}
   0x000104d0 <+4>:     add     r11, sp, #12
   0x000104d4 <+8>:     mov     r5, #3
   0x000104d8 <+12>:    mov     r3, r1
   0x000104dc <+16>:    cmp     r0, r3
   0x000104e0 <+20>:    bne     0x104ec <main+32>
   0x000104e4 <+24>:    mov     r3, #44 ; 0x2c
   0x000104e8 <+28>:    orr     r2, r3, r5
=> 0x000104ec <+32>:    mov     r3, r2
   0x000104f0 <+36>:    mov     r0, r3
   0x000104f4 <+40>:    bl      0x10468 <show_reg>
   0x000104f8 <+44>:    mov     r3, #0
   0x000104fc <+48>:    mov     r0, r3
   0x00010500 <+52>:    pop     {r4, r5, r11, pc}
End of assembler dump.
(gdb) info reg
r0             0x1      1
r1             0xf6ffef64       -150999196
r2             0xf6ffef6c       -150999188
r3             0xf6ffef64       -150999196
r4             0x10504  66820
r5             0x3      3
r6             0x10340  66368
r7             0x0      0
r8             0x0      0
r9             0x0      0
r10            0xf67fe000       -159391744
r11            0xf6ffee14       -150999532
r12            0xf6ffee90       -150999408
sp             0xf6ffee08       0xf6ffee08
lr             0xf6688d14       -160920300
pc             0x104ec  0x104ec <main+32>
cpsr           0x10     16
```

< eors > -xor

```
jhb@onestar:~/my/Homework/hanbyuljung/class/class_45_me$ qemu-arm-static -L /usr/arm-lin
x-gnueabi ./a.out
00000000000000000000000000001111
```

```
   0x000104cc <+0>:     push    {r4, r5, r11, lr}
   0x000104d0 <+4>:     add     r11, sp, #12
   0x000104d4 <+8>:     mov     r0, #0
   0x000104d8 <+12>:    mov     r1, #0
   0x000104dc <+16>:    mov     r2, #0
   0x000104e0 <+20>:    mov     r3, #0
   0x000104e4 <+24>:    mov     r4, #0
   0x000104e8 <+28>:    mov     r5, #0
   0x000104ec <+32>:    mov     r2, r0
   0x000104f0 <+36>:    mov     r3, r1
   0x000104f4 <+40>:    cmp     r2, r3
   0x000104f8 <+44>:    bne     0x10508 <main+60>
   0x000104fc <+48>:    mov     r0, #10
=> 0x00010500 <+52>:    mov     r3, #5
   0x00010504 <+56>:    eors    r1, r3, r0
   0x00010508 <+60>:    mov     r3, r1
   0x0001050c <+64>:    mov     r0, r3
   0x00010510 <+68>:    bl      0x10468 <show_reg>
   0x00010514 <+72>:    mov     r3, #0
   0x00010518 <+76>:    mov     r0, r3
   0x0001051c <+80>:    pop     {r4, r5, r11, pc}
End of assembler dump.
(gdb) info reg
r0             0xa      10
r1             0x0      0
r2             0x0      0
r3             0x0      0
r4             0x0      0
r5             0x0      0
r6             0x10340  66368
r7             0x0      0
r8             0x0      0
r9             0x0      0
r10            0xf67fe000       -159391744
r11            0xf6ffee14       -150999532
r12            0xf6ffee90       -150999408
sp             0xf6ffee08       0xf6ffee08
lr             0xf6688d14       -160920300
pc             0x10500  0x10500 <main+52>
cpsr           0x60000010       1610612752
```

```
(gdb) info reg
r0             0xa      10
r1             0xf      15
r2             0x0      0
r3             0x5      5
r4             0x0      0
r5             0x0      0
r6             0x10340  66368
r7             0x0      0
r8             0x0      0
r9             0x0      0
r10            0xf67fe000       -159391744
r11            0xf6ffee14       -150999532
r12            0xf6ffee90       -150999408
sp             0xf6ffee08       0xf6ffee08
lr             0xf6688d14       -160920300
pc             0x10508  0x10508 <main+60>
cpsr           0x20000010       536870928
```