2.4 Signed and Unsigned Numbers

정수 (integer)

Base-n Numbers (n진수)

- i번째 자리수가 d이면 d x Base^i 를 의미 (i는 0부터 시작)
- d 의 범위는 0 ~ Base - 1

$138_{10} = 1 \times 10^2 + 3 \times 10^1 + 8 \times 10^0$ when $n = 10$

Base-*2* Numbers = Binary Numbers (2진수) when *n=2*

- *i*번째 자리수가 *d*이면 $d \times 2^i$ 를 의미 (*i*는 0부터 시작)
- *d* 의 범위는 0 ~ 1

$$1011_2 = 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$$
$$= 8 + 0 + 2 + 1 = 11_{10}$$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1

MSB(Most Significant Bit)

LSB(Least Significant Bit)

Unsigned Binary Numbers

$$(x31 \times 2^{31}) + (x30 \times 2^{30}) + (x29 \times 2^{29}) + \ldots + (x1 \times 2^{1}) + (x0 \times 2^{0})$$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1

MSB(Most Significant Bit)

LSB(Least Significant Bit)

unsigned 32-bit 2진수

$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two} = 0_{ten}$

$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_{two} = 1_{ten}$

$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_{two} = 2_{ten}$

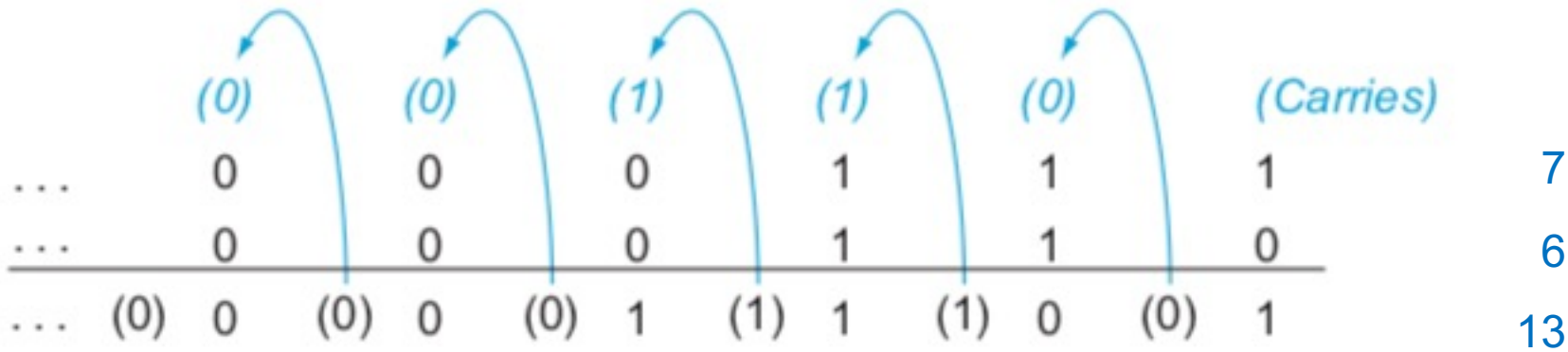
... ...

$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_{two} = 4{,}294{,}967{,}293_{ten}$

$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_{two} = 4{,}294{,}967{,}294_{ten}$

$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two} = 4{,}294{,}967{,}295_{ten}$

unsigned binary number 의 덧셈 : 7+6



FP Regs | Int Regs [16] | Data | Text

Int Regs [16]

```
PC        = 400030
EPC       = 0
Cause     = 0
BadVAddr  = 0
Status    = 3000ff10

HI        = 0
LO        = 0

R0  [r0]  = 0
R1  [at]  = 0
R2  [v0]  = c
R3  [v1]  = 0
R4  [a0]  = 3
R5  [a1]  = 7ffffe30
R6  [a2]  = 7ffffe40
R7  [a3]  = 0
R8  [t0]  = 7
R9  [t1]  = 6
R10 [t2]  = d
R11 [t3]  = 0
```

Text

```
                                User Text Segment [00400000]..[00440000
[00400000] 8fa40000  lw $4, 0($29)              ; 183: lw $a0 0($sp) # a
[00400004] 27a50004  addiu $5, $29, 4           ; 184: addiu $a1 $sp 4 #
[00400008] 24a60004  addiu $6, $5, 4            ; 185: addiu $a2 $a1 4 #
[0040000c] 00041080  sll $2, $4, 2              ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2            ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]      ; 188: jal main
[00400018] 00000000  nop                        ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10             ; 191: li $v0 10
[00400020] 0000000c  syscall                    ; 192: syscall # syscall
[00400024] 20080007  addi $8, $0, 7             ; 4: addi $t0, $0, 7
[00400028] 20090006  addi $9, $0, 6             ; 5: addi $t1, $0, 6
[0040002c] 01095020  add $10, $8, $9            ; 6: add $t2, $t0, $t1

                                Kernel Text Segment [80000000]..[8001000
[80000180] 0001d821  addu $27, $0, $1           ; 90: move $k1 $at # Sav
[80000184] 3c019000  lui $1, -28672             ; 92: sw $v0 s1 # Not re
[80000188] ac220200  sw $2, 512($1)
[8000018c] 3c019000  lui $1, -28672             ; 93: sw $a0 s2 # But we
[80000190] ac240204  sw $4, 516($1)
[80000194] 401a6800  mfc0 $26, $13              ; 95: mfc0 $k0 $13 # Cau
```

2.4 Signed and Unsigned Numbers

Category	Instruction	Example		Meaning	Comments
	add	add	$s1,$s2,$s3	$s1 = $s2 + $s3	Three operands; overflow detected
	subtract	sub	$s1,$s2,$s3	$s1 = $s2 – $s3	Three operands; overflow detected
	add immediate	addi	$s1,$s2,100	$s1 = $s2 + 100	+ constant; overflow detected
Arithmetic	add unsigned	addu	$s1,$s2,$s3	$s1 = $s2 + $s3	Three operands; overflow undetected
	subtract unsigned	subu	$s1,$s2,$s3	$s1 = $s2 – $s3	Three operands; overflow undetected
	add immediate unsigned	addiu	$s1,$s2,100	$s1 = $s2 + 100	+ constant; overflow undetected
	move from coprocessor register	mfc0	$s1,$epc	$s1 = $epc	Copy Exception PC + special regs
	multiply	mult	$s2,$s3	Hi, Lo = $s2 × $s3	64-bit signed product in Hi, Lo
	multiply unsigned	multu	$s2,$s3	Hi, Lo = $s2 × $s3	64-bit unsigned product in Hi, Lo
	divide	div	$s2,$s3	Lo = $s2 / $s3, Hi = $s2 mod $s3	Lo = quotient, Hi = remainder
	divide unsigned	divu	$s2,$s3	Lo = $s2 / $s3, Hi = $s2 mod $s3	Unsigned quotient and remainder
	move from Hi	mfhi	$s1	$s1 = Hi	Used to get copy of Hi
	move from Lo	mflo	$s1	$s1 = Lo	Used to get copy of Lo

$t0 = $t1 – 1 은 MIPS instruction 으로 어떻게?

there is no ‘subi’ instruction.

```
addi $t0, $t1, -1
```

MIPS source file

```
.text
.globl main
main:
    addi $t1, $0, -2
    addi $t0, $t1, -1
```

FP Regs | Int Regs [16]

Int Regs [16]

```
PC        = 0
EPC       = 0
Cause     = 0
BadVAddr  = 0
Status    = 3000ff10

HI        = 0
LO        = 0

R0  [r0]  = 0
R1  [at]  = 0
R2  [v0]  = 0
R3  [v1]  = 0
R4  [a0]  = 0
R5  [a1]  = 0
R6  [a2]  = 7ffffe48
R7  [a3]  = 0
R8  [t0]  = 0
R9  [t1]  = 0
R10 [t2]  = 0
R11 [t3]  = 0
R12 [t4]  = 0
R13 [t5]  = 0
R14 [t6]  = 0
```

Data | Text

Text

```
                                    User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)              ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4           ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4            ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2              ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2            ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]      ; 188: jal main
[00400018] 00000000  nop                        ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10             ; 191: li $v0 10
[00400020] 0000000c  syscall                    ; 192: syscall # syscall 10 (ex
[00400024] 2009fffe  addi $9, $0, -2            ; 4: addi $t1, $0, -2
[00400028] 2128ffff  addi $8, $9, -1            ; 5: addi $t0, $t1, -1

                                    Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821  addu $27, $0, $1           ; 90: move $k1 $at # Save $at
[80000184] 3c019000  lui $1, -28672             ; 92: sw $v0 s1 # Not re-entran
[80000188] ac220200  sw $2, 512($1)
[8000018c] 3c019000  lui $1, -28672             ; 93: sw $a0 s2 # But we need t
[80000190] ac240204  sw $4, 516($1)
[80000194] 401a6800  mfc0 $26, $13              ; 95: mfc0 $k0 $13 # Cause regi
[80000198] 001a2082  srl $4, $26, 2             ; 96: srl $a0 $k0 2 # Extract E
[8000019c] 3084001f  andi $4, $4, 31            ; 97: andi $a0 $a0 0x1f
[800001a0] 34020004  ori $2, $0, 4              ; 101: li $v0 4 # syscall 4 (pr
[800001a4] 3c049000  lui $4, -28672 [  m1 ]     ; 102: la $a0   m1
```

FP Regs | Int Regs [16] | Data | Text

Int Regs [16]

```
PC       = 400028
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 3000ff10

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = c
R3  [v1] = 0
R4  [a0] = 3
R5  [a1] = 7ffffdf0
R6  [a2] = 7ffffe00
R7  [a3] = 0
R8  [t0] = 0
R9  [t1] = fffffffe
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
```

Text

```
                                        User Text Segment [00400000]..[00
[00400000] 8fa40000  lw $4, 0($29)              ; 183: lw $a0 0($sp) #
[00400004] 27a50004  addiu $5, $29, 4           ; 184: addiu $a1 $sp 4
[00400008] 24a60004  addiu $6, $5, 4            ; 185: addiu $a2 $a1 4
[0040000c] 00041080  sll $2, $4, 2              ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2            ; 187: addu $a2 $a2 $v
[00400014] 0c100009  jal 0x00400024 [main]      ; 188: jal main
[00400018] 00000000  nop                        ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10             ; 191: li $v0 10
[00400020] 0000000c  syscall                    ; 192: syscall # sysca
[00400024] 2009fffe  addi $9, $0, -2            ; 4: addi $t1, $0, -2
[00400028] 2128ffff  addi $8, $9, -1            ; 5: addi $t0, $t1, -1

                                        Kernel Text Segment [80000000]..[8
[80000180] 0001d821  addu $27, $0, $1           ; 90: move $k1 $at # S
[80000184] 3c019000  lui $1, -28672             ; 92: sw $v0 s1 # Not
[80000188] ac220200  sw $2, 512($1)
[8000018c] 3c019000  lui $1, -28672             ; 93: sw $a0 s2 # But
[80000190] ac240204  sw $4, 516($1)
[80000194] 401a6800  mfc0 $26, $13              ; 95: mfc0 $k0 $13 # C
[80000198] 001a2082  srl $4, $26, 2             ; 96: srl $a0 $k0 2 #
[8000019c] 3084001f  andi $4, $4, 31            ; 97: andi $a0 $a0 0x1
[800001a0] 34020004  ori $2, $0, 4              ; 101: li $v0 4 # sysc
[800001a4] 3c049000  lui $4, -28672 [  m1 ]     ; 102: la $a0   m1
```

How to represent negative numbers

```
addi $t0, $t1, -1
```

unsigned 32-bit 2진수

$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two} = 0_{ten}$

$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_{two} = 1_{ten}$

$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_{two} = 2_{ten}$

... ...

$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_{two} = 4{,}294{,}967{,}293_{ten}$

$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_{two} = 4{,}294{,}967{,}294_{ten}$

$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two} = 4{,}294{,}967{,}295_{ten}$

signed 32-bit 2진수 in 2's complement representation

sign bit

$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two} = 0_{ten}$
$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_{two} = 1_{ten}$
$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_{two} = 2_{ten}$
... ...
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_{two} = 2{,}147{,}483{,}645_{ten}$
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_{two} = 2{,}147{,}483{,}646_{ten}$
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two} = 2{,}147{,}483{,}647_{ten}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two} = -2{,}147{,}483{,}648_{ten}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_{two} = -2{,}147{,}483{,}647_{ten}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_{two} = -2{,}147{,}483{,}646_{ten}$
... ...
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_{two} = -3_{ten}$
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_{two} = -2_{ten}$
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two} = -1_{ten}$

Signed Binary Numbers

$$(x31 \times -2^{31}) + (x30 \times 2^{30}) + (x29 \times 2^{29}) + \ldots + (x1 \times 2^{1}) + (x0 \times 2^{0})$$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1

MSB(Most Significant Bit)

LSB(Least Significant Bit)

Unsigned Binary Numbers

$$(x31 \times 2^{31}) + (x30 \times 2^{30}) + (x29 \times 2^{29}) + \ldots + (x1 \times 2^{1}) + (x0 \times 2^{0})$$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1

MSB(Most Significant Bit)

LSB(Least Significant Bit)

Signed Binary Number (2의 보수 표현)

What is the decimal value of this 32-bit two's complement number?

$$1111\ \ 1111\ \ 1111\ \ 1111\ \ 1111\ \ 1111\ \ 1111\ \ 1100_{two}$$

Substituting the number's bit values into the formula above:

$$(1 \times -2^{31}) + (1 \times 2^{30}) + (1 \times 2^{29}) + \ldots + (1 \times 2^{2}) + (0 \times 2^{1}) + (0 \times 2^{0})$$
$$= -2^{31} + 2^{30} + 2^{29} + \ldots + 2^{2} + 0 + 0$$

$$= -2^{31} + \sum_{i=2}^{30} 2^{i} = -2^{31} + 4(2^{29} - 1)$$

$$\sum_{k=1}^{n} a_k = \frac{a(1-r^n)}{1-r}$$

$$= -2^{31} + 2^{31} - 4$$
$$= -4$$

** 2진수를 10진수로 바꾸는 것은 사람이 이해하기 위한 것이지 컴퓨터는 10진수로 바꾸어 볼 필요가 없습니다.

signed 32-bit 2진수 in 2's complement representation

-2 를 표현하려면 +2 의 2의 보수를 만든다.

1) +2 를 2진수로 표현한다. → a

$$2_{ten} = 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_{two}$$

2) a 의 1의 보수를 만든다. → b

$$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_{two}$$

3) b 에 1을 더한다. → c 는 -2

$$\begin{array}{ll} & 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_{two} \\ + & 1_{two} \\ \hline = & 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_{two} \\ = & -2_{ten} \end{array}$$

signed 32-bit 2진수 in 2's complement representation

a=1111 1111 1111 1111 1111 1111 1111 1111_2
은 십진수로 얼마인가?

1) 음수이면 a 의 1의 보수를 만든다 → b

b=0000 0000 0000 0000 0000 0000 0000 0000

2) b 에 1을 더한다. → c 가 a 의 절대값이다.

c=0000 0000 0000 0000 0000 0000 0000 0001

3) a 는 십진수로 –c
그러므로 a 는 십진수로 -1

*) 양수이면?

d=0111 1111 1111 1111 1111 1111 1111 1111_2

$= -\mathbf{0} \times 2^{31} + \sum_{i=0}^{30} 2^i = 2^{\mathbf{31}} - 1 = 2147483647$

Register 값들을 hex 로 보여줌

FP Regs | Int Regs [16] | Data | Text

Int Regs [16]

```
PC       = 400030
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 3000ff10

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = c
R3  [v1] = 0
R4  [a0] = 3
R5  [a1] = 7ffffe30
R6  [a2] = 7ffffe40
R7  [a3] = 0
R8  [t0] = fffffffe
R9  [t1] = ffffffff
R10 [t2] = ffffffff
R11 [t3] = 0
```

Text

```
                    User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)              ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4           ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4            ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2              ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2            ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]      ; 188: jal main
[00400018] 00000000  nop                        ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10             ; 191: li $v0 10
[00400020] 0000000c  syscall                    ; 192: syscall # syscall 10 (e
[00400024] 2008fffe  addi $8, $0, -2            ; 4: addi $t0, $0, -2
[00400028] 2009ffff  addi $9, $0, -1            ; 5: addi $t1, $0, 0xffffffff
[0040002c] 200affff  addi $10, $0, -1           ; 6: addi $t2, $0, -1

                    Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821  addu $27, $0, $1           ; 90: move $k1 $at # Save $at
[80000184] 3c019000  lui $1, -28672             ; 92: sw $v0 s1 # Not re-entra
[80000188] ac220200  sw $2, 512($1)
[8000018c] 3c019000  lui $1, -28672             ; 93: sw $a0 s2 # But we need
[80000190] ac240204  sw $4, 516($1)
[80000194] 401a6800  mfc0 $26, $13              ; 95: mfc0 $k0 $13 # Cause reg
```

Register 값들을 decimal 로 보여줌

FP Regs | Int Regs [10] | Data | Text

Int Regs [10]

```
PC       = 4194352
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 805371664

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 12
R3  [v1] = 0
R4  [a0] = 3
R5  [a1] = 2147483184
R6  [a2] = 2147483200
R7  [a3] = 0
R8  [t0] = -2
R9  [t1] = -1
R10 [t2] = -1
R11 [t3] = 0
```

Text

```
                                User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)              ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4           ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4            ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2              ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2            ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]      ; 188: jal main
[00400018] 00000000  nop                        ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10             ; 191: li $v0 10
[00400020] 0000000c  syscall                    ; 192: syscall # syscall 10 (e
[00400024] 2008fffe  addi $8, $0, -2            ; 4: addi $t0, $0, -2
[00400028] 2009ffff  addi $9, $0, -1            ; 5: addi $t1, $0, 0xffffffff
[0040002c] 200affff  addi $10, $0, -1           ; 6: addi $t2, $0, -1

                                Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821  addu $27, $0, $1           ; 90: move $k1 $at # Save $at
[80000184] 3c019000  lui $1, -28672             ; 92: sw $v0 s1 # Not re-entra
[80000188] ac220200  sw $2, 512($1)
[8000018c] 3c019000  lui $1, -28672             ; 93: sw $a0 s2 # But we need
[80000190] ac240204  sw $4, 516($1)
[80000194] 401a6800  mfc0 $26, $13              ; 95: mfc0 $k0 $13 # Cause reg
```

Register 값들을 binary 로 보여줌

FP Regs | Int Regs [2]

Int Regs [2]

```
PC       = 10000000000000000110000
EPC      = 0
Cause    = 0
BadVAddr = 0
Status   = 11000000000000111111100010000

HI       = 0
LO       = 0

R0  [r0] = 0
R1  [at] = 0
R2  [v0] = 1100
R3  [v1] = 0
R4  [a0] = 11
R5  [a1] = 11111111111111111111111000110000
R6  [a2] = 11111111111111111111111001000000
R7  [a3] = 0
R8  [t0] = 11111111111111111111111111111110
R9  [t1] = 11111111111111111111111111111111
R10 [t2] = 11111111111111111111111111111111
R11 [t3] = 0
```

Data | Text

Text

```
                    User Text Segment [00400000]..[004
[00400000] 8fa40000  lw $4, 0($29)            ; 183: lw
[00400004] 27a50004  addiu $5, $29, 4         ; 184: ad
[00400008] 24a60004  addiu $6, $5, 4          ; 185: ad
[0040000c] 00041080  sll $2, $4, 2            ; 186: sl
[00400010] 00c23021  addu $6, $6, $2          ; 187: ad
[00400014] 0c100009  jal 0x00400024 [main]    ; 188: ja
[00400018] 00000000  nop                      ; 189: no
[0040001c] 3402000a  ori $2, $0, 10           ; 191: li
[00400020] 0000000c  syscall                  ; 192: sy
[00400024] 2008fffe  addi $8, $0, -2          ; 4: addi
[00400028] 2009ffff  addi $9, $0, -1          ; 5: addi
[0040002c] 200affff  addi $10, $0, -1         ; 6: addi

                   Kernel Text Segment [80000000]..[80
[80000180] 0001d821  addu $27, $0, $1         ; 90: mov
[80000184] 3c019000  lui $1, -28672           ; 92: sw
and we can't trust $sp
[80000188] ac220200  sw $2, 512($1)
[8000018c] 3c019000  lui $1, -28672           ; 93: sw
use these registers
```