정수 곱셈, 나눗셈

Category	Instruction	Example	Meaning	Comments
Arithmetic	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
	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

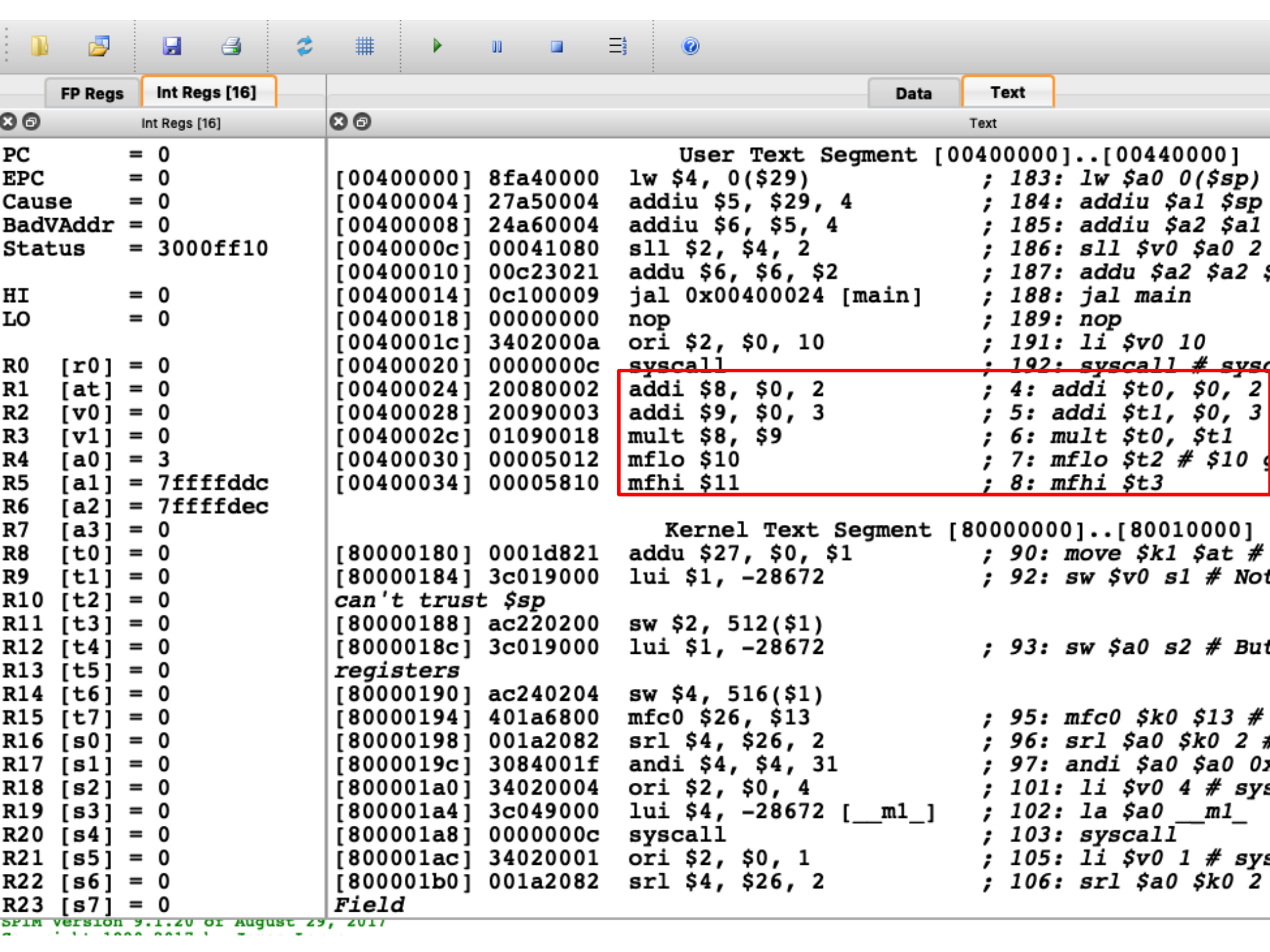
MIPS Integer Multiplication

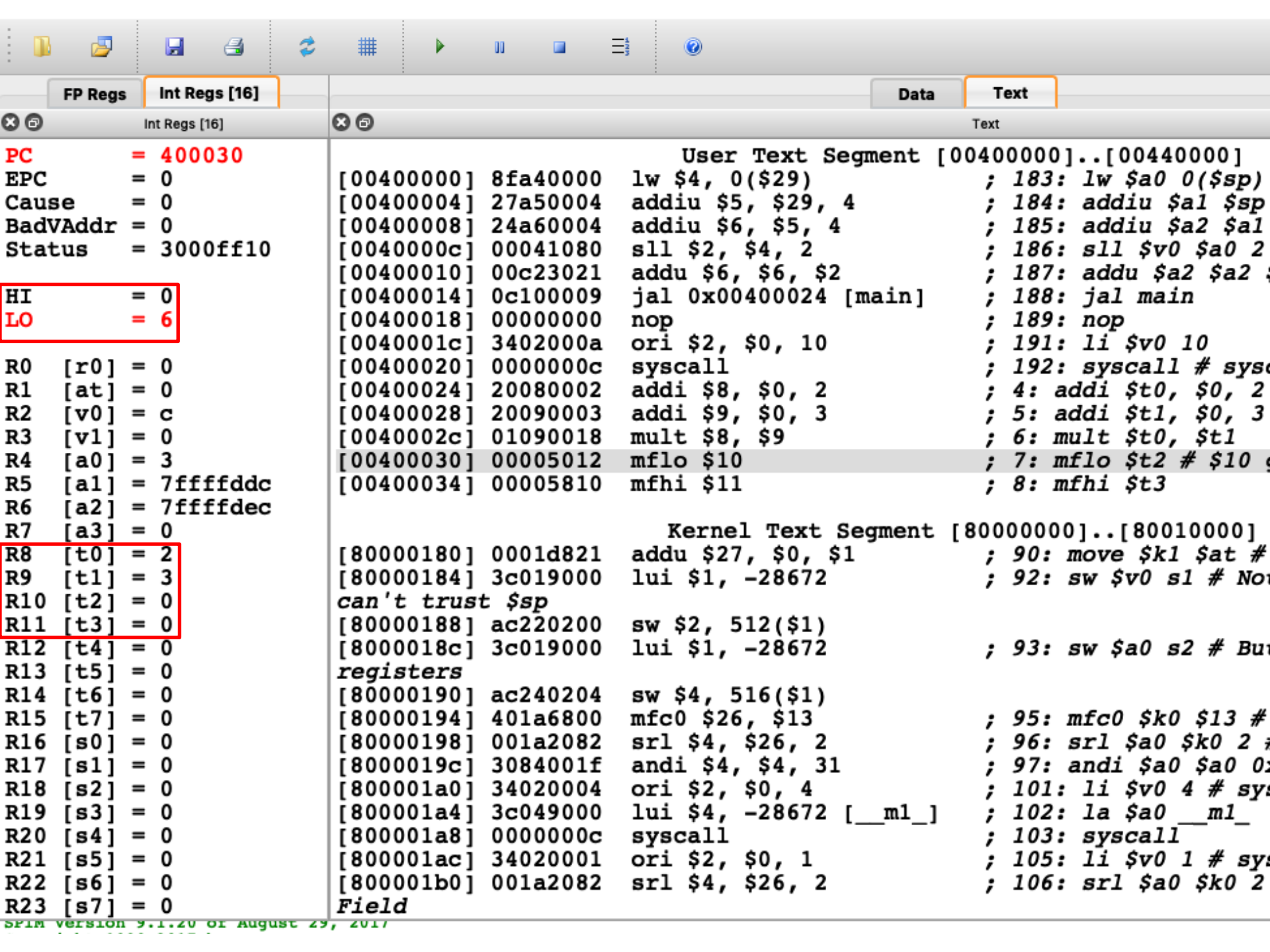
- Instructions require 2 operands
  - `mult rs, rt  /  multu rs, rt`
    - 64-bit product in HI/LO
- Two 32-bit registers for product
  - HI: most-significant 32 bits of product
  - LO: least-significant 32-bits of product
  - `mfhi rd  /  mflo rd`
    - Move from HI/LO to rd
- No overflow check for mult / multu
  - Can test HI value to see if product overflows 32 bits

program 2 x 3 in MIPS assembly

product 는 $t2 에 저장 **mult.s**

```
.text
.globl main
main:
    addi $t0, $0, 2 # $8 gets 2
    addi $t1, $0, 3 # $9 gets 3
    mult $t0, $t1
    mflo $t2 # $10 gets 2x3
    mfhi $t3
```


FP Regs
Int Regs [16]
Data
Text
Int Regs [16]
Text
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] = 3
R5 [a1] = 7fffddc
R6 [a2] = 7fffdec
R7 [a3] = 0
R8 [t0] = 0
R9 [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
User Text Segment [00400000]..[00440000]
[00400000] 8fa40000 lw $4, 0($29) ; 183: lw $a0 0($sp)
[00400004] 27a50004 addiu $5, $29, 4 ; 184: addiu $a1 $sp
[00400008] 24a60004 addiu $6, $5, 4 ; 185: addiu $a2 $a1
[0040000c] 00041080 sll $2, $4, 2 ; 186: sll $v0 $a0 2
[00400010] 00c23021 addu $6, $6, $2 ; 187: addu $a2 $a2
[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
[00400024] 20080002 addi $8, $0, 2 ; 4: addi $t0, $0, 2
[00400028] 20090003 addi $9, $0, 3 ; 5: addi $t1, $0, 3
[0040002c] 01090018 mult $8, $9 ; 6: mult $t0, $t1
[00400030] 00005012 mflo $10 ; 7: mflo $t2 # $10
[00400034] 00005810 mfhi $11 ; 8: mfhi $t3
Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821 addu $27, $0, $1 ; 90: move $k1 $at #
[80000184] 3c019000 lui $1, -28672 ; 92: sw $v0 s1 # Not
can't trust $sp
[80000188] ac220200 sw $2, 512($1)
[8000018c] 3c019000 lui $1, -28672 ; 93: sw $a0 s2 # But
registers
[80000190] ac240204 sw $4, 516($1)
[80000194] 401a6800 mfc0 $26, $13 ; 95: mfc0 $k0 $13 #
[80000198] 001a2082 srl $4, $26, 2 ; 96: srl $a0 $k0 2
[8000019c] 3084001f andi $4, $4, 31 ; 97: andi $a0 $a0 0x
[800001a0] 34020004 ori $2, $0, 4 ; 101: li $v0 4 # sys
[800001a4] 3c049000 lui $4, -28672 [__m1_] ; 102: la $a0 __m1_
[800001a8] 0000000c syscall ; 103: syscall
[800001ac] 34020001 ori $2, $0, 1 ; 105: li $v0 1 # sys
[800001b0] 001a2082 srl $4, $26, 2 ; 106: srl $a0 $k0 2
Field
SPIM Version 9.1.20 of August 29, 2017


FP Regs
Int Regs [16]
Data
Text
Int Regs [16]
Text
PC = 400030
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000ff10
HI = 0
LO = 6
R0 [r0] = 0
R1 [at] = 0
R2 [v0] = c
R3 [v1] = 0
R4 [a0] = 3
R5 [a1] = 7fffddc
R6 [a2] = 7fffdec
R7 [a3] = 0
R8 [t0] = 2
R9 [t1] = 3
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
User Text Segment [00400000]..[00440000]
[00400000] 8fa40000 lw $4, 0($29) ; 183: lw $a0 0($sp)
[00400004] 27a50004 addiu $5, $29, 4 ; 184: addiu $a1 $sp
[00400008] 24a60004 addiu $6, $5, 4 ; 185: addiu $a2 $a1
[0040000c] 00041080 sll $2, $4, 2 ; 186: sll $v0 $a0 2
[00400010] 00c23021 addu $6, $6, $2 ; 187: addu $a2 $a2
[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 # sys
[00400024] 20080002 addi $8, $0, 2 ; 4: addi $t0, $0, 2
[00400028] 20090003 addi $9, $0, 3 ; 5: addi $t1, $0, 3
[0040002c] 01090018 mult $8, $9 ; 6: mult $t0, $t1
[00400030] 00005012 mflo $10 ; 7: mflo $t2 # $10
[00400034] 00005810 mfhi $11 ; 8: mfhi $t3
Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821 addu $27, $0, $1 ; 90: move $k1 $at #
[80000184] 3c019000 lui $1, -28672 ; 92: sw $v0 s1 # No
can't trust $sp
[80000188] ac220200 sw $2, 512($1)
[8000018c] 3c019000 lui $1, -28672 ; 93: sw $a0 s2 # Bu
registers
[80000190] ac240204 sw $4, 516($1)
[80000194] 401a6800 mfc0 $26, $13 ; 95: mfc0 $k0 $13 #
[80000198] 001a2082 srl $4, $26, 2 ; 96: srl $a0 $k0 2
[8000019c] 3084001f andi $4, $4, 31 ; 97: andi $a0 $a0 0
[800001a0] 34020004 ori $2, $0, 4 ; 101: li $v0 4 # sy
[800001a4] 3c049000 lui $4, -28672 [__m1_] ; 102: la $a0 __m1_
[800001a8] 0000000c syscall ; 103: syscall
[800001ac] 34020001 ori $2, $0, 1 ; 105: li $v0 1 # sy
[800001b0] 001a2082 srl $4, $26, 2 ; 106: srl $a0 $k0 2
Field
SPIM Version 9.1.20 of August 29, 2017

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

Int Regs [16]

```
PC          = 400038
EPC         = 0
Cause       = 0
BadVAddr    = 0
Status      = 3000ff10

HI          = 0
LO          = 6

R0   [r0]   = 0
R1   [at]   = 0
R2   [v0]   = c
R3   [v1]   = 0
R4   [a0]   = 3
R5   [a1]   = 7fffddc
R6   [a2]   = 7fffdec
R7   [a3]   = 0
R8   [t0]   = 2
R9   [t1]   = 3
R10  [t2]   = 6
R11  [t3]   = 0
R12  [t4]   = 0
R13  [t5]   = 0
R14  [t6]   = 0
R15  [t7]   = 0
R16  [s0]   = 0
R17  [s1]   = 0
R18  [s2]   = 0
R19  [s3]   = 0
R20  [s4]   = 0
R21  [s5]   = 0
R22  [s6]   = 0
R23  [s7]   = 0
```

Text

```
                    User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)              ; 183: lw $a0 0($sp)
[00400004] 27a50004  addiu $5, $29, 4          ; 184: addiu $a1 $sp
[00400008] 24a60004  addiu $6, $5, 4           ; 185: addiu $a2 $a1
[0040000c] 00041080  sll $2, $4, 2             ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2           ; 187: addu $a2 $a2 $
[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 # sysc
[00400024] 20080002  addi $8, $0, 2            ; 4: addi $t0, $0, 2
[00400028] 20090003  addi $9, $0, 3            ; 5: addi $t1, $0, 3
[0040002c] 01090018  mult $8, $9               ; 6: mult $t0, $t1
[00400030] 00005012  mflo $10                  ; 7: mflo $t2 # $10 g
[00400034] 00005810  mfhi $11                  ; 8: mfhi $t3

                    Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821  addu $27, $0, $1          ; 90: move $k1 $at #
[80000184] 3c019000  lui $1, -28672            ; 92: sw $v0 s1 # Not
can't trust $sp
[80000188] ac220200  sw $2, 512($1)
[8000018c] 3c019000  lui $1, -28672            ; 93: sw $a0 s2 # But
registers
[80000190] ac240204  sw $4, 516($1)
[80000194] 401a6800  mfc0 $26, $13             ; 95: mfc0 $k0 $13 #
[80000198] 001a2082  srl $4, $26, 2            ; 96: srl $a0 $k0 2 #
[8000019c] 3084001f  andi $4, $4, 31           ; 97: andi $a0 $a0 0x
[800001a0] 34020004  ori $2, $0, 4             ; 101: li $v0 4 # sys
[800001a4] 3c049000  lui $4, -28672 [__m1_]    ; 102: la $a0 __m1_
[800001a8] 0000000c  syscall                   ; 103: syscall
[800001ac] 34020001  ori $2, $0, 1             ; 105: li $v0 1 # sys
[800001b0] 001a2082  srl $4, $26, 2            ; 106: srl $a0 $k0 2
Field
```

SPIM Version 9.1.20 of August 29, 2017

MIPS Integer Division

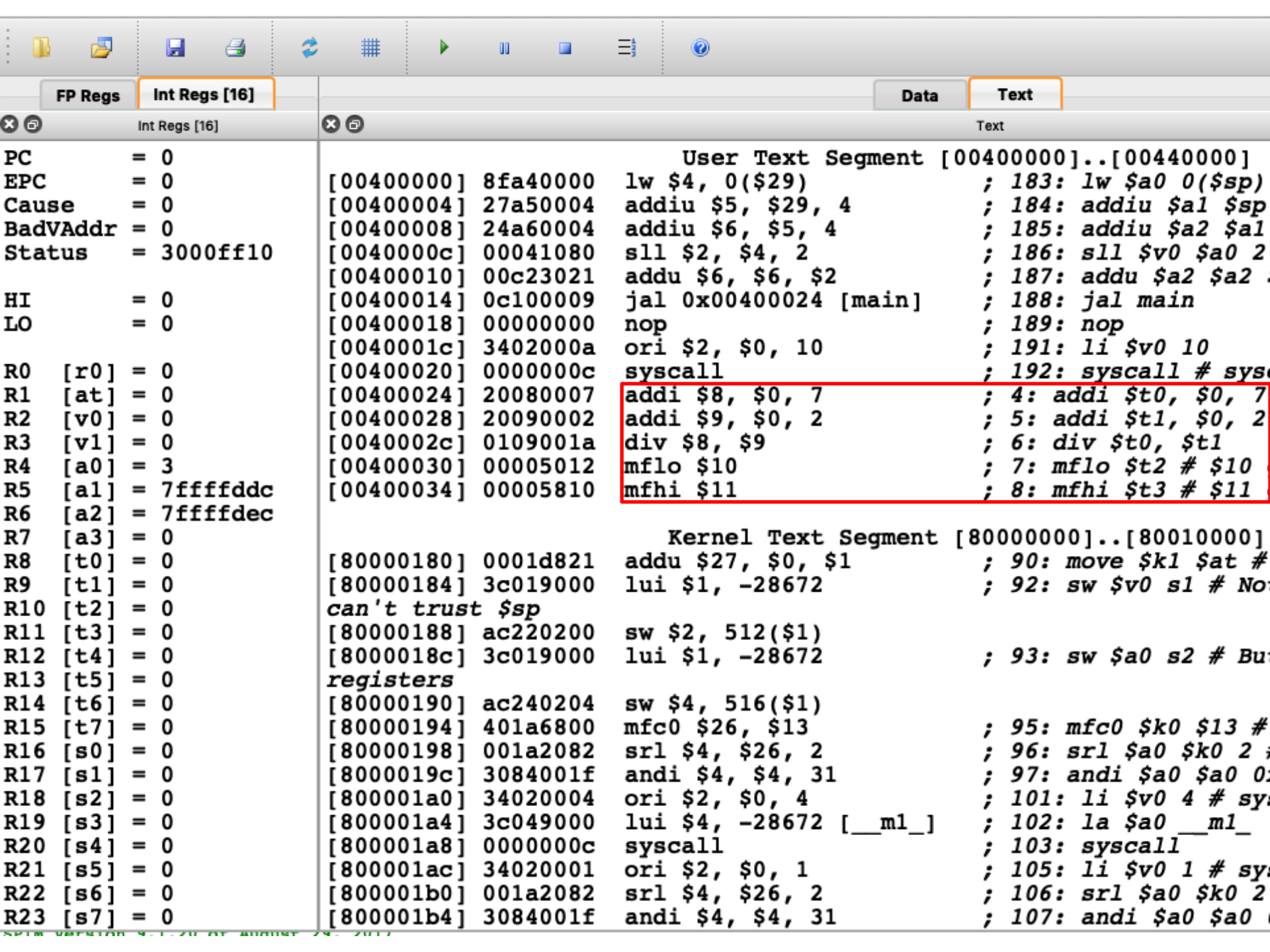
- Instructions require 2 operands
  - `div rs, rt` / `divu rs, rt`
  - No overflow or divide-by-0 checking
    - Software must perform checks if required
- Use HI/LO registers for result
  - HI: 32-bit remainder
  - LO: 32-bit quotient
  - Use `mfhi`, `mflo` to access result

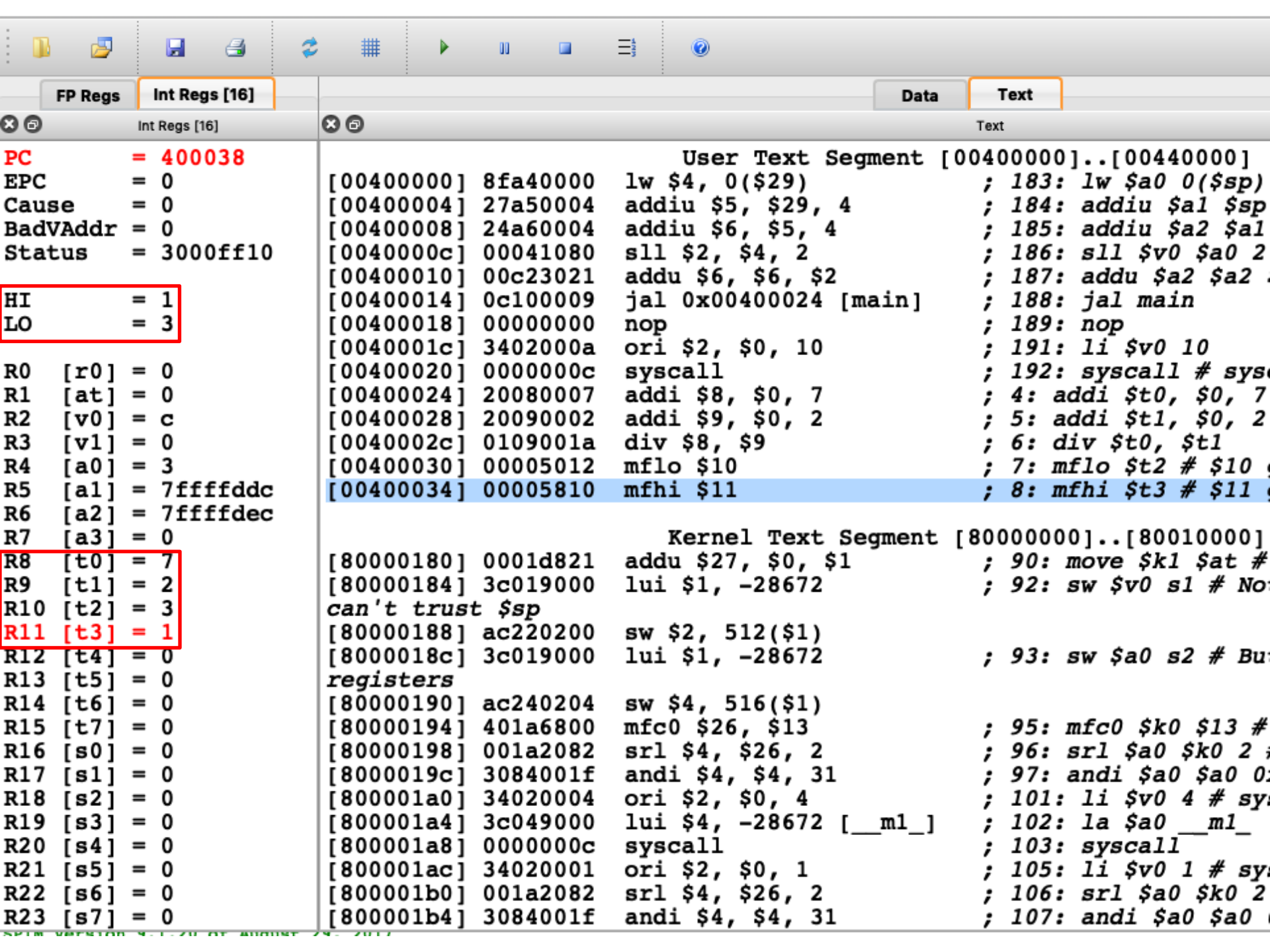
program 7 / 2 in MIPS assembly

product 는 $t2 에, remainder 는 $t3에 저장

div.s

```
.text
.globl main
main:
    addi $t0, $0, 7 # $8 gets 7
    addi $t1, $0, 2 # $9 gets 2
    div  $t0, $t1
    mflo $t2 # $10 gets 7/2
    mfhi $t3 # $11 gets 7%2
```


FP Regs
Int Regs [16]
Data
Text
Int Regs [16]
Text
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] = 3
R5 [a1] = 7fffddc
R6 [a2] = 7fffdec
R7 [a3] = 0
R8 [t0] = 0
R9 [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
User Text Segment [00400000]..[00440000]
[00400000] 8fa40000 lw $4, 0($29) ; 183: lw $a0 0($sp)
[00400004] 27a50004 addiu $5, $29, 4 ; 184: addiu $a1 $sp
[00400008] 24a60004 addiu $6, $5, 4 ; 185: addiu $a2 $a1
[0040000c] 00041080 sll $2, $4, 2 ; 186: sll $v0 $a0 2
[00400010] 00c23021 addu $6, $6, $2 ; 187: addu $a2 $a2
[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 # sys
[00400024] 20080007 addi $8, $0, 7 ; 4: addi $t0, $0, 7
[00400028] 20090002 addi $9, $0, 2 ; 5: addi $t1, $0, 2
[0040002c] 0109001a div $8, $9 ; 6: div $t0, $t1
[00400030] 00005012 mflo $10 ; 7: mflo $t2 # $10
[00400034] 00005810 mfhi $11 ; 8: mfhi $t3 # $11
Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821 addu $27, $0, $1 ; 90: move $k1 $at #
[80000184] 3c019000 lui $1, -28672 ; 92: sw $v0 s1 # No
can't trust $sp
[80000188] ac220200 sw $2, 512($1)
[8000018c] 3c019000 lui $1, -28672 ; 93: sw $a0 s2 # Bu
registers
[80000190] ac240204 sw $4, 516($1)
[80000194] 401a6800 mfc0 $26, $13 ; 95: mfc0 $k0 $13 #
[80000198] 001a2082 srl $4, $26, 2 ; 96: srl $a0 $k0 2
[8000019c] 3084001f andi $4, $4, 31 ; 97: andi $a0 $a0 0
[800001a0] 34020004 ori $2, $0, 4 ; 101: li $v0 4 # sy
[800001a4] 3c049000 lui $4, -28672 [__m1_] ; 102: la $a0 __m1_
[800001a8] 0000000c syscall ; 103: syscall
[800001ac] 34020001 ori $2, $0, 1 ; 105: li $v0 1 # sy
[800001b0] 001a2082 srl $4, $26, 2 ; 106: srl $a0 $k0 2
[800001b4] 3084001f andi $4, $4, 31 ; 107: andi $a0 $a0


FP Regs
Int Regs [16]
Data
Text
Int Regs [16]
Text
PC = 400038
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000ff10
HI = 1
LO = 3
R0 [r0] = 0
R1 [at] = 0
R2 [v0] = c
R3 [v1] = 0
R4 [a0] = 3
R5 [a1] = 7fffddc
R6 [a2] = 7fffdec
R7 [a3] = 0
R8 [t0] = 7
R9 [t1] = 2
R10 [t2] = 3
R11 [t3] = 1
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
User Text Segment [00400000]..[00440000]
[00400000] 8fa40000 lw $4, 0($29) ; 183: lw $a0 0($sp)
[00400004] 27a50004 addiu $5, $29, 4 ; 184: addiu $a1 $sp
[00400008] 24a60004 addiu $6, $5, 4 ; 185: addiu $a2 $a1
[0040000c] 00041080 sll $2, $4, 2 ; 186: sll $v0 $a0 2
[00400010] 00c23021 addu $6, $6, $2 ; 187: addu $a2 $a2
[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 # sys
[00400024] 20080007 addi $8, $0, 7 ; 4: addi $t0, $0, 7
[00400028] 20090002 addi $9, $0, 2 ; 5: addi $t1, $0, 2
[0040002c] 0109001a div $8, $9 ; 6: div $t0, $t1
[00400030] 00005012 mflo $10 ; 7: mflo $t2 # $10
[00400034] 00005810 mfhi $11 ; 8: mfhi $t3 # $11
Kernel Text Segment [80000000]..[80010000]
[80000180] 0001d821 addu $27, $0, $1 ; 90: move $k1 $at #
[80000184] 3c019000 lui $1, -28672 ; 92: sw $v0 s1 # No
can't trust $sp
[80000188] ac220200 sw $2, 512($1)
[8000018c] 3c019000 lui $1, -28672 ; 93: sw $a0 s2 # Bu
registers
[80000190] ac240204 sw $4, 516($1)
[80000194] 401a6800 mfc0 $26, $13 ; 95: mfc0 $k0 $13 #
[80000198] 001a2082 srl $4, $26, 2 ; 96: srl $a0 $k0 2
[8000019c] 3084001f andi $4, $4, 31 ; 97: andi $a0 $a0 0
[800001a0] 34020004 ori $2, $0, 4 ; 101: li $v0 4 # sy
[800001a4] 3c049000 lui $4, -28672 [__m1_] ; 102: la $a0 __m1_
[800001a8] 0000000c syscall ; 103: syscall
[800001ac] 34020001 ori $2, $0, 1 ; 105: li $v0 1 # sy
[800001b0] 001a2082 srl $4, $26, 2 ; 106: srl $a0 $k0 2
[800001b4] 3084001f andi $4, $4, 31 ; 107: andi $a0 $a0

정수 곱셈, 나눗셈

Category	Instruction	Example	Meaning	Comments
Arithmetic	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
	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