

Tutorial- Introduction to MIPS



What is QtSpim?



- ◆ Spim is a self-contained simulator that runs MIPS32 programs. (MIPS-Microprocessor without Interlocked Pipeline Stages)
- ◆ QTSpim is a new user interface for Spim built on the Qt UI framework.
- ◆ Download QTSpim from:-
<https://sourceforge.net/projects/spimsimulator/files/latest/download>

Memory

- ◆ Memory is Byte Addressable
- ◆ MIPS as simulated in QTSpim is little-endian

	Low address				High address			
Address	0	1	2	3	4	5	6	7
Little-endian	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Big-endian	Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
Memory content	0x11	0x22	0x33	0x44	0x55	0x66	0x77	0x88
64 bit value on Little-endian				64 bit value on Big-endian				
0x8877665544332211				0x1122334455667788				

QTSpm Program Formats

- ◆ For comment line we used '#' symbol.
- ◆ Assembly directive informs the assembler for data declarations and also where is the text and code. It begins with a '.'
- ◆ The general format for data declaration is :-

`<variableName>: .<dataType> <initialValue>`

QTSpim: Loading and Executing Programs

◆ Start QTSpim

◆ The initial
QtSpim screen
will appear as
shown.

```
QtSpim
File Simulator Registers Text Segment Data Segment Window Help

FP Regs Int Regs [16] Data Text

Int Regs [16]
PC = 0
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000fff10

HI = 0
LO = 0

R0 [r0] = 0
R1 [at] = 0
R2 [v0] = 0
R3 [v1] = 0
R4 [a0] = 1
R5 [a1] = 7ffff144
R6 [a2] = 7ffff14c
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
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0

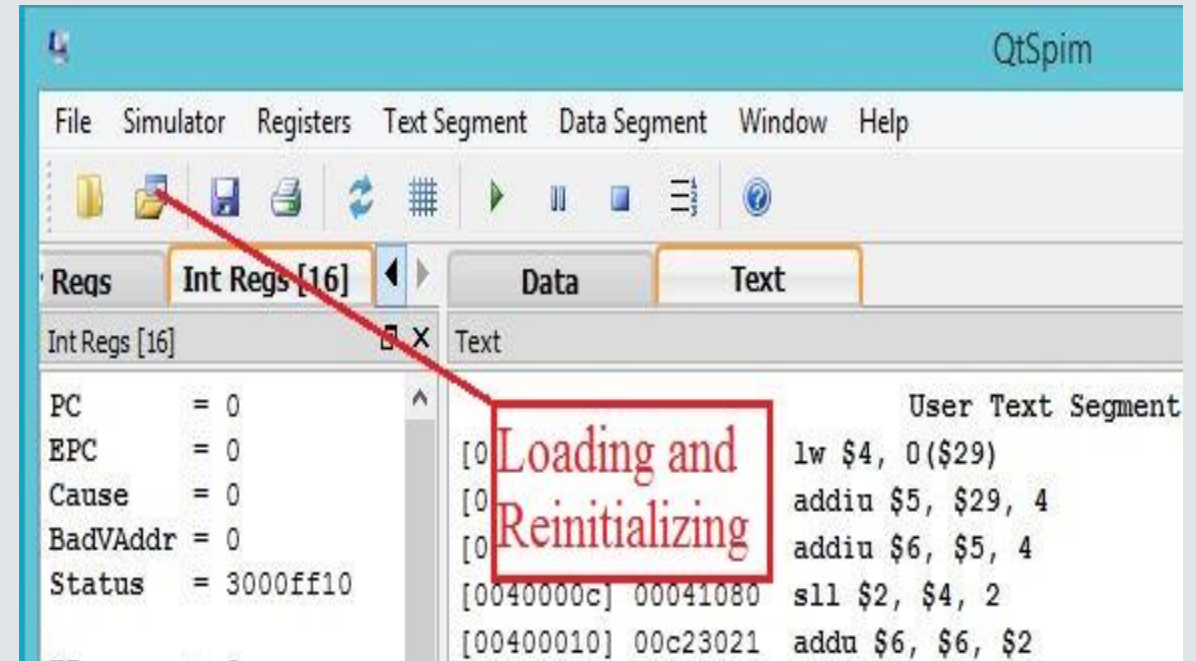
Text
Ctrl+N User Text Segment [00400000]..[00440000]
[00400000] 8fa40000 lw $4, 0($29) ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004 addiu $5, $29, 4 ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004 addiu $6, $5, 4 ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080 sll $2, $4, 2 ; 186: sll $v0 $a0 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 (exit)
[00400024] 3410000a ori $16, $0, 10 ; 16: li $s0, 10
[00400028] 34020004 ori $2, $0, 4 ; 19: li $v0, 4
[0040002c] 3c041001 lui $4, 4097 [prompt] ; 20: la $a0, prompt
[00400030] 0000000c syscall ; 21: syscall
[00400034] 34020005 ori $2, $0, 5 ; 24: li $v0, 5
[00400038] 0000000c syscall ; 25: syscall
[0040003c] 00028021 addu $16, $0, $2 ; 26: move $s0, $v0
[00400040] 34020004 ori $2, $0, 4 ; 31: li $v0, 4
[00400044] 3c011001 lui $1, 4097 [str1] ; 32: la $a0, str1
[00400048] 34240016 ori $4, $1, 22 [str1] ; 33: syscall
[0040004c] 0000000c syscall ; 36: li $v0, 1
[00400050] 34020001 ori $2, $0, 1 ; 37: move $a0, $s0
[00400054] 00102021 addu $4, $0, $16 ; 38: syscall
[00400058] 0000000c syscall ; 41: li $v0, 4
[0040005c] 34020004 ori $2, $0, 4 ; 42: la $a0, newline
[00400060] 3c011001 lui $1, 4097 [newline] ; 43: syscall
[00400064] 34240026 ori $4, $1, 38 [newline] ; 46: sub $s0, $s0, 1
[00400068] 0000000c syscall ; 47: bgez $s0, loop
[0040006c] 2210ffff addi $16, $16, -1 ; 50: li $v0, 4
[00400070] 0601ffff bgez $16 -48 [loop-0x00400070]; 51: la $a0, bye
[00400074] 34020004 ori $2, $0, 4 ; 52: syscall
[00400078] 3c011001 lui $1, 4097 [bye] ; 53: jr $ra # retrain to caller
[0040007c] 34240028 ori $4, $1, 40 [bye]
[00400080] 0000000c syscall
[00400084] 03e00008 jr $31

SPIM Version 9.1.9 of January 19, 2013
Copyright 1990-2012, James R. Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.

Running
```

Load & Reinitialize

- ◆ To load and reinitialize click on the button highlighted.
- ◆ Once selected, a standard open file dialog box will be displayed. Select the program to be executed



Text Segment

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] = 1
R5 [a1] = 7ffffa4c
R6 [a2] = 7ffffa54
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
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0

User Text Segment [00400000]..[00400000]

```
100400000: 8fa40000 lw $4, 0($29) ; 183: lw $a0 0($s0) # argc
100400004: 27a30004 addiu $5, $29, 4 ; 184: addiu $a1 $s0 4 # argv
100400008: 25a60004 addiu $6, $5, 4 ; 185: addiu $a2 $a1 4 # envp
10040000c: 00041080 sll $2, $4, 2 ; 186: sll $v0 $a0 2
100400010: 00c23021 addu $6, $6, $2 ; 187: addu $a2 $a2 $v0
100400014: 0c100009 jal 0x00400024 [main] ; 188: jal main
100400018: 00000000 nop ; 189: nop
10040001c: 3402000a ori $2, $0, 10 ; 191: li $v0 10
100400020: 0000000c syscall ; 192: syscall # syscall 10 (exit)
100400024: 3c011001 lui $1, 4097 [hdr] ; 43: la $a0, hdr
100400028: 34240040 ori $4, $1, 64 [hdr]
10040002c: 34020004 ori $2, $0, 4 ; 46: li $v0, 4
100400030: 0000000c syscall ; 47: syscall # print header
100400034: 3c081001 lui $8, 4097 [array] ; 56: la $t0, array # set $t0 addr of array
100400038: 3c011001 lui $1, 4097 ; 57: lw $t1, len # set $t1 to length
10040003c: 8c29003c lw $9, 60($1) ; 59: lw $s2, ($t0) # set min, $t2 to array[0]
100400040: 8d120000 lw $18, 0($8) ; 60: lw $s3, ($t0) # set max, $t3 to array[0]
100400044: 8d130000 lw $19, 0($8) ; 62: lw $t4, ($t0) # get array[1]
100400048: 8d0c0000 lw $12, 0($8) ; 64: bge $t4, $s2, NotMin # is new min?
10040004c: 0192082a slt $1, $12, $18 ; 65: move $s2, $t4 # set new min
100400050: 10200002 beq $1, $0, 8 [NotMin-0x00400050] ; 67: ble $t4, $s3, NotMax # is new max?
100400054: 000c9021 addu $18, $0, $12 ; 68: move $s3, $t4 # set new max
100400058: 026c082a slt $1, $19, $12 ; 71: sub $t1, $t1, 1 # decrement counter
10040005c: 10200002 beq $1, $0, 8 [NotMax-0x0040005c] ; 72: addu $t0, $t0, 4 # increment addr by word
100400060: 000c9821 addu $19, $0, $12 ; 80: la $a0, al_msg
100400064: 2129ffff addi $9, $9, -1 ; 81: li $v0, 4
100400068: 25080004 addiu $8, $8, 4 ; 82: syscall # print "min = "
10040006c: 1520ffe7 bne $9, $0, -36 [loop-0x0040006c] ; 84: move $a0, $s2
100400070: 3c011001 lui $1, 4097 [al_msg] ; 85: li $v0, 1
100400074: 34240069 ori $4, $1, 105 [al_msg] ; 86: syscall # print min
100400078: 34020004 ori $2, $0, 4 ; 88: la $a0, new_ln # print a newline
10040007c: 0000000c syscall
100400080: 00122021 addu $4, $0, $18
100400084: 34020001 ori $2, $0, 1
100400088: 0000000c syscall
10040008c: 3c011001 lui $1, 4097 [new_ln]
```

Copyright 1990-2012, James R. L. All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.

Addresses

OpCodes

Bare-Instructions

Psuedo-Instructions

Data Segment

FP Regs Int Regs [16] Data Text

Int Regs [16]

PC = 400070
EPC = 400070
Cause = 24
BadVAddr = 0
Status = 3000ff10

HI = 0
LO = 0

R0 [r0] = 0
R1 [at] = 0
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 10010040
R5 [a1] = 7ffffa4c
R6 [a2] = 7ffffa54
R7 [a3] = 0
R8 [t0] = 1001003c
R9 [t1] = 0
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = d
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [a0] = 0
R17 [a1] = 0
R18 [a2] = 7
R19 [a3] = 3d
R20 [a4] = 0
R21 [a5] = 0
R22 [a6] = 0
R23 [a7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0

User data segment [10000000]..[10040000]

[10000000]..[1000ffff]	00000000	00000022	00000010	0000003d
[10010000]	0000000d	00000018	0000003a	0000000b
[10010010]	0000001c	00000029	00000013	00000007
[10010020]	0000001a	00000028	00000013	0000000f
[10010030]	00000026	0000000c	0000000d	0000000f
[10010040]	6178450a	656c706d	6f727020	6d617267	. E x a m p l e
[10010050]	206f7420	646e6966	78616d20	646e6120	. t o f i n d m a x a n d
[10010060]	6e696d20	0a000a0a	6e696d00	00203d20	. m i n . . . m i n = .
[10010070]	2078616d	0000203d	00000000	00000000	. m a x =
[10010080]..[1003ffff]	00000000				

User Stack [7ffffa48]..[80000000]

[7ffffa48]	00000001	7ffffad0		
[7ffffa50]	00000000	7ffffa09	7ffffa01	7ffffa04
[7ffffa60]	7ffffb06	7ffffb0f	7ffffb0e	7ffffb1a
[7ffffa70]	7ffffb00	7ffffb01	7ffffb0b	7ffffb00
[7ffffa80]	7ffffb0d	7ffffb07	7ffffb0b	7ffffb02
[7ffffa90]	7ffffb01	7ffffb01	7ffffb01	7ffffb05
[7ffffaa0]	7ffffb02	7ffffb03	7ffffb06	7ffffb0b
[7ffffab0]	7ffffb09	7ffffb0b	7ffffb0f	7ffffb03
[7ffffac0]	7ffffb09	7ffffb0e	7ffffb0d	7ffffb0a
[7ffffad0]	00000000	00000000	00000000	6d6f682f
[7ffffae0]	64652f65	6f72442f	786f6270	6e69752f
[7ffffaf0]	696d2f76	742f7370	726f7475	2f6e6169
[7ffffb00]	74737361	73612e30	4947006d	414c5f4f
[7ffffb10]	48434e55	445f4445	544b5345	465f504f
[7ffffb20]	55454c49	3d444950	39333038	4f494700
[7ffffb30]	55414c5f	4548434e	45445f44	4f544b53
[7ffffb40]	49465f50	2f727375	72616873	72616873
[7ffffb50]	70612f65	63696e70	6f697461	712f736e
[7ffffb60]	69707374	65642e6d	6f746b73	50470070
[7ffffb70]	47415f47	5f544e45	4f464e49	6d742f3d
[7ffffb80]	656b2f70	6e697279	6d662d67	4f774a56
[7ffffb90]	6770672f	313e003a	47445800	5255435f
[7ffffba0]	544e4552	534e005f	504f544b	696e553d

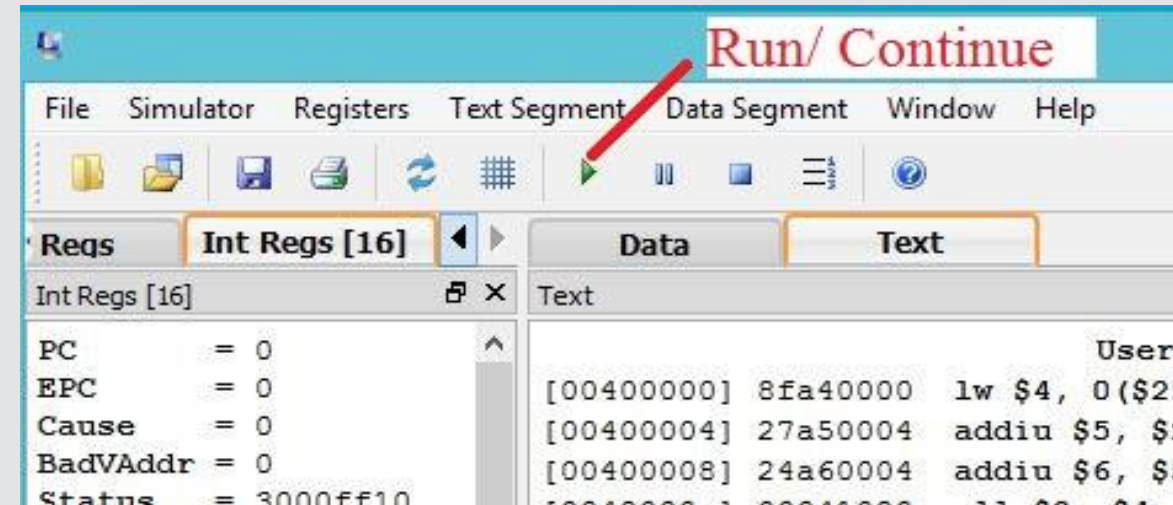
Copyright 1990-2012, James R. ...
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for full copyright notice.

Running

Addresses Data (Hex Representation) Data (ASCII Representation)

Execution

- ◆ To run the program click on the button highlighted
- ◆ If a program performs input and/or output, it will be directed to the Console window as shown.



Register Pane

QtSpim

File Simulator Registers Text Segment Data Segment Window Help

Reqs Int Regs [16] Data Text

Int Regs [16]

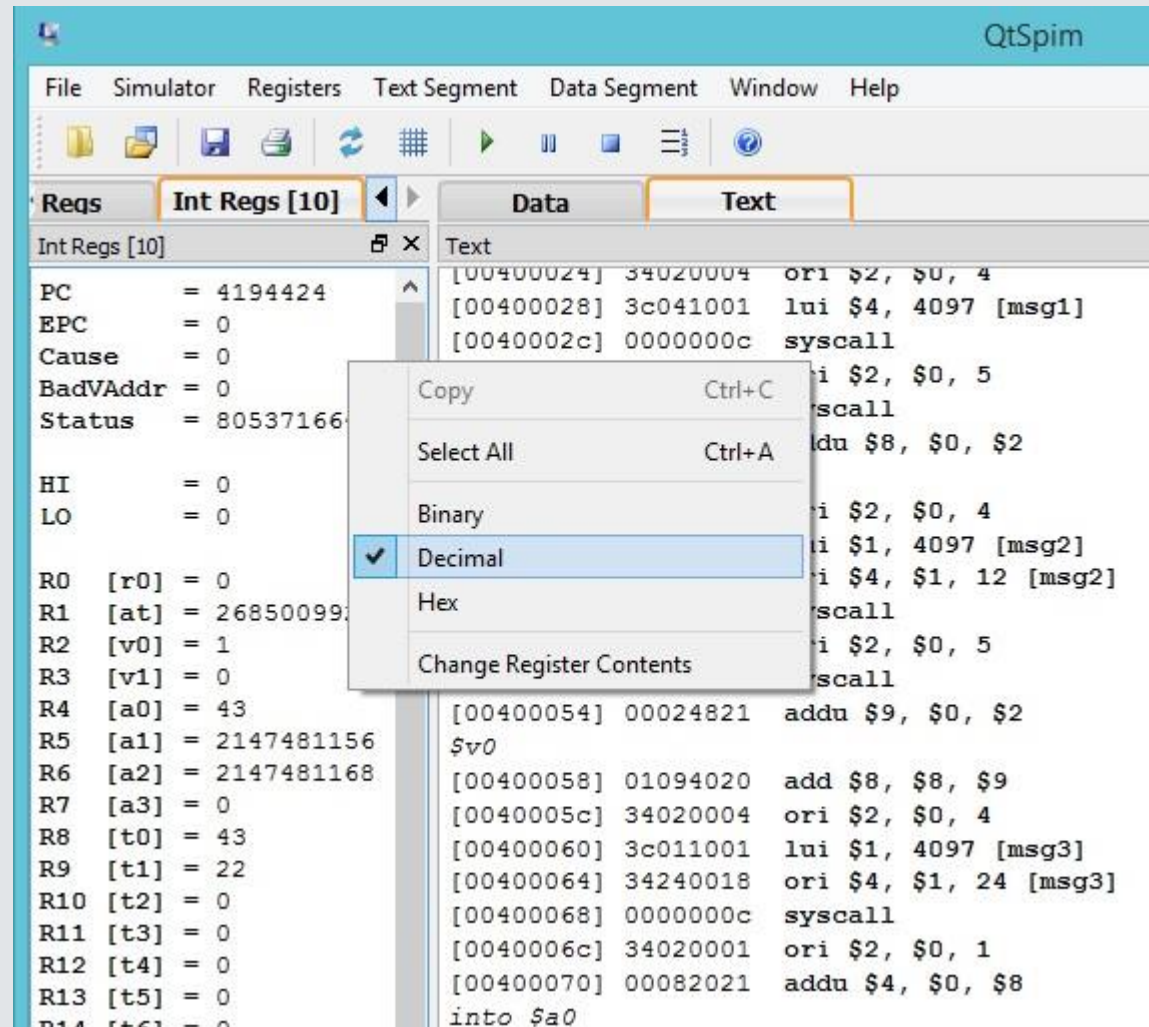
PC = 400078
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000ff10
HI = 0
LO = 0
R0 [r0] = 0
R1 [at] = 10010000
R2 [v0] = 1
R3 [v1] = 0
R4 [a0] = 2b
R5 [a1] = 7ffff644
R6 [a2] = 7ffff650
R7 [a3] = 0
R8 [t0] = 2b
R9 [t1] = 16
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = 0
R17 [s1] = 0
R18 [s2] = 0
R19 [s3] = 0
R20 [s4] = 0
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0

Text

```
[00400024] 34020004 ori $2, $0, 4 ; 12: li $v0,4 # print_string syscall code = 4
[00400028] 3c041001 lui $4, 4097 [msg1] ; 13: la $a0, msg1 # load the address of msg
[0040002c] 0000000c syscall ; 14: syscall
[00400030] 34020005 ori $2, $0, 5 ; 17: li $v0,5 # read_int syscall code = 5
[00400034] 0000000c syscall ; 18: syscall
[00400038] 00024021 addu $8, $0, $2 ; 19: move $t0,$v0 # syscall results returned in $v0
[0040003c] 34020004 ori $2, $0, 4 ; 22: li $v0,4 # print_string syscall code = 4
[00400040] 3c011001 lui $1, 4097 [msg2] ; 23: la $a0, msg2 # load the address of msg2
[00400044] 3424000c ori $4, $1, 12 [msg2] ; 
[00400048] 0000000c syscall ; 24: syscall
[0040004c] 34020005 ori $2, $0, 5 ; 27: li $v0,5 # read_int syscall code = 5
[00400050] 0000000c syscall ; 28: syscall
[00400054] 00024821 addu $9, $0, $2 ; 29: move $t1,$v0 # syscall results returned in $v0
[00400058] 01094020 add $8, $8, $9 ; 32: add $t0, $t0, $t1 # A = A + B
[0040005c] 34020004 ori $2, $0, 4 ; 35: li $v0, 4
[00400060] 3c011001 lui $1, 4097 [msg3] ; 36: la $a0, msg3
[00400064] 34240018 ori $4, $1, 24 [msg3] ; 
[00400068] 0000000c syscall ; 37: syscall
[0040006c] 34020001 ori $2, $0, 1 ; 40: li $v0,1 # print_int syscall code = 1
[00400070] 00082021 addu $4, $0, $8 ; 41: move $a0, $t0 # int to print must be loaded into $a0
[00400074] 0000000c syscall ; 42: syscall
[00400078] 34020004 ori $2, $0, 4 ; 45: li $v0,4 # print_string syscall code = 4
[0040007c] 3c011001 lui $1, 4097 [newline] ; 46: la $a0, newline
[00400080] 34240021 ori $4, $1, 33 [newline] ; 
[00400084] 0000000c syscall ; 47: syscall
[00400088] 3402000a ori $2, $0, 10 ; 49: li $v0,10 # exit
[0040008c] 0000000c syscall ; 50: syscall

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 $1 # Not re-entrant and we can't trust $sp
```

Decimal Representation



Directive

- ◆ **.DATA** directive
 - ◆ Defines the data segment of a program containing data
 - ◆ The program's variables should be defined under this directive
 - ◆ Assembler will allocate and initialize the storage of variables
- ◆ **.TEXT** directive
 - ◆ Defines the code segment of a program containing instructions
- ◆ **.GLOBL** directive
 - ◆ Declares a symbol as global
 - ◆ Global symbols can be referenced from other files
 - ◆ We use this directive to declare *main* procedure of a program

Sample Program

```
# Program to print a string on the screen
```

```
.data
```

```
msg:  .asciiiz  "This is CO Lab"
```

```
.text
```

```
.globl  main
```

```
main:    li $v0, 4                #System call code for print string
         la $a0, msg1             #Address of NULL terminated string
         syscall

         li $v0, 10              #System call code for exit
         syscall
```

Basic Printing and Reading Exercise

- ◆ Write a MIPS program to print Computer Organization on the console
- ◆ Write a MIPS program to read an integer from the keyboard and display in the screen.
- ◆ Write a MIPS program to read a string from the keyboard and display in the screen.

Load and Store Exercise

- ◆ Write a MIPS program to load two numbers from memory and add them.
- ◆ Write a MIPS program to load two numbers stored in consecutive memory locations and add them. Store the result in the next memory location.



Thank You