

計算機組織與結構 – HOMEWORK1

撰寫MIPS程式 (共題，100分，滿分100分)

請於2025/12/02前上傳至M數位園區作業區繳交

請安裝QtSpim (<http://spimsimulator.sourceforge.net/>) 模擬器，並請詳細參考課本第二章及附錄A的介紹，於QtSpim模擬器環境下，撰寫一完整的MIPS核心指令集版本的程式。(需貼完整程式碼，截圖呈現結果並文字說明。)

(1)實作第二章2.7小節範例if-then-else，請自行完成變數設定，觀察暫存器及記憶體狀態並說明程式之運作。(50分)

(2)實作第二章2.7小節範例while迴圈，請自行完成變數設定，觀察暫存器及記憶體狀態並說明程式之運作。(50分)

Asian Edition

計算機組織與設計

Computer Organization and Design



QtSpim

- *spim* is a simulator that runs MIPS32 programs
- It's been around for more than 20 years (improving over time).
- QtSpim is a new interface for *spim* built on the Qt UI framework which supports various platforms (Windows, Mac, Linux)
- It reads and executes assembly language programs.
- It contains a simple debugger



Start SPIM

Int Regs [16]

Register	Value
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]	= 0
R5 [a1]	= 0
R6 [a2]	= 7ffffa40
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

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] 0c000000 jal 0x00000000 [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)

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
[80000188] ac220200 sw $2, 512($1) ; 93: sw $a0 $2 # But we need to use these registers
[8000018c] 3c019000 lui $1, -28672 ; 95: mfc0 $k0 $13 # Cause register
[80000190] ac240204 sw $4, 516($1) ; 96: srl $a0 $k0 2 # Extract ExcCode Field
[80000194] 401a6800 mfc0 $26, $13 ; 97: andi $a0 $a0 0x1f
[80000198] 001a2082 srl $4, $26, 2 ; 101: li $v0 4 # syscall 4 (print_str)
[8000019c] 3084001f andi $4, $4, 31 ; 102: la $a0 __mi_
[800001a0] 34020004 ori $2, $0, 4 ; 103: syscall
[800001a4] 3c049000 lui $4, -28672 [__mi_] ; 105: li $v0 1 # syscall 1 (print_int)
[800001a8] 0000000c syscall ; 106: srl $a0 $k0 2 # Extract ExcCode Field
[800001ac] 34020001 ori $2, $0, 1 ; 107: andi $a0 $a0 0x1f
[800001b0] 001a2082 srl $4, $26, 2 ; 108: syscall
[800001b4] 3084001f andi $4, $4, 31 ; 110: li $v0 4 # syscall 4 (print_str)
[800001b8] 0000000c syscall ; 111: andi $a0 $k0 0x3c
[800001bc] 34020004 ori $2, $0, 4 ; 112: lw $a0 __excp($a0)
[800001c0] 3344003c andi $4, $26, 60 ; 113: nop
[800001c4] 3c019000 lui $1, -28672 ; 114: syscall
[800001c8] 00240821 addu $1, $1, $4 ; 116: bne $k0 0x18 ok no # Bad PC exception requires special checks
[800001cc] 8c240180 lw $4, 384($1)
[800001d0] 00000000 nop
[800001d4] 0000000c syscall
[800001d8] 34020004 ori $2, $0, 4
```

Messages

QtSpim version 1.1.0 of January 2, 2012.
Copyright 1990-2010, James R. Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.

Load Program

QtSpim

File Simulator Registers Text Segment Data Segment Window Help

Load File
Recent Files
Reinitialize and Load File
Save Log File
Print
Exit

load your assembly code

HI = 0
LO = 0

R0 [r0] = 0
R1 [at] = 0
R2 [v0] = a
R3 [v1] = 0
R4 [a0] = ffffffffef1
R5 [a1] = 7ffffa00
R6 [a2] = 7ffffa08
R7 [a3] = 0
R8 [t0] = ffffffffef1
R9 [t1] = 10
R10 [t2] = 0
R11 [t3] = 0
R12 [t4] = 0
R13 [t5] = 0
R14 [t6] = 0
R15 [t7] = 0
R16 [s0] = ffffffffef1
R17 [s1] = 5
R18 [s2] = ffffffffefec
R19 [s3] = d
R20 [s4] = 3
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 400018
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0

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] 001fb821 addu $23, $0, $31 ; 3: addu $s7, $0, $ra #save the return address in a global register
[00400028] 20110005 addi $17, $0, 5 ; 5: addi $s1, $0, 5 #g = 5
[0040002c] 2012ffec addi $18, $0, -20 ; 6: addi $s2, $0, -20 #h = -20
[00400030] 2013000d addi $19, $0, 13 ; 7: addi $s3, $0, 13 #i = 13
[00400034] 20140003 addi $20, $0, 3 ; 8: addi $s4, $0, 3 #j = 3
[00400038] 02324020 add $8, $17, $18 ; 9: add $t0, $s1, $s2 #register $t0 contains g + h
[0040003c] 02744820 add $9, $19, $20 ; 10: add $t1, $s3, $s4 #register $t1 contains i + j
[00400040] 01098022 sub $16, $8, $9 ; 11: sub $s0, $t0, $t1 #f = (g + h) - (i + j)
[00400044] 34020004 ori $2, $0, 4 ; 17: li $v0, 4 #print_str (system call 4)
[00400048] 3c041001 lui $4, 4097 [message] ; 18: la $a0, message # takes the address of string as an argument
[0040004c] 0000000c syscall ; 19: syscall
[00400050] 34020001 ori $2, $0, 1 ; 21: li $v0, 1 #print_int (system call 1)
[00400054] 00102020 add $4, $0, $16 ; 22: add $a0, $0, $s0 #put value to print in $a0
[00400058] 0000000c syscall ; 23: syscall
[0040005c] 0017f821 addu $31, $0, $23 ; 26: addu $ra, $0, $s7 #restore the return address
[00400060] 03e00008 jr $31 ; 27: jr $ra #return to the main program
[00400064] 00000020 add $0, $0, $0 ; 28: add $0, $0, $0 #nop
```

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
[80000188] ac220200 sw $2, 512($1) ; 
[8000018c] 3c019000 lui $1, -28672 ; 93: sw $a0 $2 # But we need to use these registers
[80000190] ac240204 sw $4, 516($1) ; 
[80000194] 401c6800 mfc0 $26, $12 ; 95: mfc0 $k0 $12 # Cause register
```

SPIM Version 9.1.5 of January 2, 2012
Copyright 1990-2010, James R. Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.

Execute Program

The image shows the QtSpim MIPS simulator interface. The 'Int Regs [16]' window on the left displays the current state of the registers. The 'Text' window on the right shows the assembly code for the 'User Text Segment'. Two red arrows point to specific elements: one to the 'PC' register value (40002c) and another to the instruction at address 0040002c.

Int Regs [16]

Register	Value
PC	40002c
EPC	0
Cause	0
BadVAddr	0
Status	3000ff10
HI	0
LO	0
R0 [r0]	0
R1 [at]	0
R2 [v0]	4
R3 [v1]	0
R4 [a0]	1
R5 [a1]	7ffff638
R6 [a2]	7ffff640
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]	5
R18 [s2]	0
R19 [s3]	0
R20 [s4]	0
R21 [s5]	0
R22 [s6]	0
R23 [s7]	400018
R24 [t8]	0
R25 [t9]	0
R26 [k0]	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 (exit)
[00400024] 001fb821 addu $23, $0, $31 ; 3: addu $s7, $0, $ra #save the return address in a global register
[00400028] 20110005 addi $17, $0, 5 ; 5: addi $s1, $0, 5 #g = 5
[0040002c] 2012ffec addi $18, $0, -20 ; 6: addi $s2, $0, -20 #h = -20
[00400030] 2013000d addi $19, $0, 13 ; 7: addi $s3, $0, 13 #i = 13
[00400034] 20140003 addi $20, $0, 3 ; 8: addi $s4, $0, 3 #j = 3
[00400038] 02324020 add $8, $17, $18 ; 9: add $t0, $s1, $s2 #register $t0 contains g + h
[0040003c] 02744820 add $9, $19, $20 ; 10: add $t1, $s3, $s4 #register $t1 contains i + j
[00400040] 01098022 sub $16, $8, $9 ; 11: sub $s0, $t0, $t1 #f = (g + h) - (i + j)
[00400044] 34020004 ori $2, $0, 4 ; 17: li $v0, 4 #print_str (system call 4)
[00400048] 3c041001 lui $4, 4097 [message] ; 18: la $a0, message # takes the address of string as an argument
[0040004c] 0000000c syscall ; 19: syscall
[00400050] 34020001 ori $2, $0, 1 ; 21: li $v0, 1 #print_int (system call 1)
[00400054] 00102020 add $4, $0, $16 ; 22: add $a0, $0, $s0 #put value to print in $a0
[00400058] 0000000c syscall ; 23: syscall
[0040005c] 0017f821 addu $31, $0, $23 ; 26: addu $ra, $0, $s7 #restore the return address
[00400060] 03e00008 jr $31 ; 27: jr $ra #return to the main program
[00400064] 00000020 add $0, $0, $0 ; 28: add $0, $0, $0 #nop
    
```

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
[80000188] ac220200 sw $2, 512($1) ; 
[8000018c] 3c019000 lui $1, -28672 ; 93: sw $a0 $2 # But we need to use these registers
[80000190] ac240204 sw $4, 516($1) ; 
[80000194] 401a5800 mfc0 $26, $12 ; 95: mfc0 $k0 $12 # Cause register
    
```

Annotations:

- Red arrow pointing to PC = 40002c: **your assembly code starts**
- Red arrow pointing to instruction at 0040002c: **current instruction**
- Red arrow pointing to R17 [s1] = 5: **register change resulting from previous instruction**

SPIM Version 9.1.4 of September 4, 2011
Copyright 1990-2010, James R. Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.

Single Step

Program data

The screenshot displays the QtSpim MIPS simulator interface. The 'Int Regs [16]' tab is selected, showing the following register values:

Register	Value
PC	400034
EPC	0
Cause	0
BadVAddr	0
Status	3000ff10
HI	0
LO	0
R0 [r0]	0
R1 [at]	0
R2 [v0]	4
R3 [v1]	0
R4 [a0]	1
R5 [a1]	7ffffa00
R6 [a2]	7ffffa08
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]	5
R18 [s2]	ffffffffffec
R19 [s3]	d
R20 [s4]	0
R21 [s5]	0
R22 [s6]	0
R23 [s7]	400018
R24 [t8]	0
R25 [t9]	0
R26 [k0]	0

The 'Data' tab is selected, showing the 'User data segment [10000000]..[10040000]'. A red box highlights the following data:

Address	Value	Comment
[10000000]..[1000ffff]	00000000	
[10010000]	6568540a 6c617620 6f206575 20662066	. The value of f
[10010010]	203a7369 00000000 00000000 00000000	is :
[10010020]..[1003ffff]	00000000	

The 'User Stack [7ffff9fc]..[80000000]' is also visible, showing memory addresses and their corresponding values.

SPIM Version 9.1.5 of January 2, 2012
 Copyright 1990-2010, James R. Larus.
 All Rights Reserved.
 SPIM is distributed under a BSD license.
 See the file README for a full copyright notice.

Set a break point

Set a break point at the conditional instruction

The screenshot shows the QtSpim MIPS simulator. The left pane displays the register file with the PC at 400050. The right pane shows the assembly code for the 'User Text Segment'. The instruction at address 00400050 is `ori $2, $0, 1`, which is highlighted in blue. A red rectangle is drawn around this instruction, and a right-click context menu is open over it. The menu contains the following options: Copy (Ctrl+C), Select All (Ctrl+A), Set Breakpoint, and Clear Breakpoint. A red arrow points to the 'Set Breakpoint' option. The assembly code continues with `syscall` and `jr $31` instructions.

Debug by stepping your code line by line

QtSpim

File Simulator Registers Text Segment Data Segment Window Help

run the program step by step (F10)

FP Regs Int Regs [16] Data Text

Int Regs [16]

PC = 400038
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000ff10
HI = 0
LO = 0
R0 [r0] = 0
R1 [at] = 0
R2 [v0] = 4
R3 [v1] = 0
R4 [a0] = 1
R5 [a1] = 7ffffa00
R6 [a2] = 7ffffa08
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] = 5
R18 [s2] = ffffffffec
R19 [s3] = d
R20 [s4] = 3
R21 [s5] = 0
R22 [s6] = 0
R23 [s7] = 400018
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0

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] 001fb821 addu $23, $0, $31 ; 3: addu $s7, $0, $ra #save the return address in a global register
[00400028] 20110005 addi $17, $0, 5 ; 5: addi $s1, $0, 5 #g = 5
[0040002c] 2012ffec addi $18, $0, -20 ; 6: addi $s2, $0, -20 #h = -20
[00400030] 2013000d addi $19, $0, 13 ; 7: addi $s3, $0, 13 #i = 13
[00400034] 20140003 addi $20, $0, 3 ; 8: addi $s4, $0, 3 #j = 3
[00400038] 02324020 add $8, $17, $18 ; 9: add $t0, $s1, $s2 #register $t0 contains g + h
[0040003c] 02744820 add $9, $19, $20 ; 10: add $t1, $s3, $s4 #register $t1 contains i + j
[00400040] 01098022 sub $16, $8, $9 ; 11: sub $s0, $t0, $t1 #f = (g + h) - (i + j)
[00400044] 34020004 ori $2, $0, 4 ; 17: li $v0, 4 #print_str (system call 4)
[00400048] 3c041001 lui $4, 4097 [message] ; 18: la $a0, message # takes the address of string as an argument
[0040004c] 0000000c syscall ; 19: syscall
[00400050] 34020001 ori $2, $0, 1 ; 21: li $v0, 1 #print_int (system call 1)
[00400054] 00102020 add $4, $0, $16 ; 22: add $a0, $0, $s0 #put value to print in $a0
[00400058] 0000000c syscall ; 23: syscall
[0040005c] 0017f821 addu $31, $0, $23 ; 26: addu $ra, $0, $s7 #restore the return address
[00400060] 03e00008 jr $31 ; 27: jr $ra #return to the main program
[00400064] 00000020 add $0, $0, $0 ; 28: add $0, $0, $0 #nop
```

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
[80000188] ac220200 sw $2, 512($1) ; 
[8000018c] 3c019000 lui $1, -28672 ; 93: sw $a0 $2 # But we need to use these registers
[80000190] ac240204 sw $4, 516($1) ; 
[80000194] 401a6800 mfc0 $26, $12 ; 95: mfc0 $k0 $12 # Cause register
```

SPIM Version 9.1.5 of January 2, 2012
Copyright 1990-2010, James R. Larus.
All Rights Reserved.
SPIM is distributed under a BSD license.
See the file README for a full copyright notice.

MIPS Hello World

Hello, World!

.data ## Data declaration section

String to be printed:

out_string: .asciiz "\nHello, World!\n"

.text ## Assembly language instructions go in text segment

main: ## Start of code section

```
li $v0, 4                    # system call code for printing string = 4
la $a0, out_string           # load address of string to be printed into $a0
syscall                    # call operating system to perform operation
                            # specified in $v0
                            # syscall takes its arguments from $a0, $a1, ...

li $v0, 10                  # terminate program
syscall
```

