

1. 撰寫 MIPS 程式 (共 2 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)

```

ASM if_then_else.asm
1 #####
2 # MIPS If-Then-Else 範例
3 #
4 # int f, g=1, h=2, i=0, j=0;
5 # if (i == j) f = g + h; else f = g - h;
6 #####
7
8      .text          # 程式碼區段開始
9      .globl main     # 宣告 main 標籤為全域
10
11 main:
12      addi $s0, $zero, 0 # $s0 (f) = 0
13      addi $s1, $zero, 1 # $s1 (g) = 1
14      addi $s2, $zero, 2 # $s2 (h) = 2
15      addi $s3, $zero, 0 # $s3 (i) = 0
16      addi $s4, $zero, 0 # $s4 (j) = 0
17      bne $s3, $s4, Else # go to Else if i != j
18      add $s0, $s1, $s2 # f = g + h (skipped if i != j)
19      j Exit           # go to Exit
20
21 Else:
22      sub $s0, $s1, $s2 # f = g - h (skipped if i = j)
23
24 Exit:
25      li $v0, 10       # 系統呼叫編號 10 (Exit)
26      syscall
27

```

QtSpim

File Simulator Registers Text Segment Data Segment Window Help

FF Regs Int Regs [16] Data Text

PC = 40004c
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000ff10

HI = 0
LO = 0

R0 [r0] = 0
R1 [at] = 0
R2 [v0] = 10
R3 [v1] = 0
R4 [a0] = 1
R5 [a1] = 7ffff528
R6 [a2] = 7ffff530
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] = 1
R18 [s2] = 2
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
R28 [gp] = 10008000
R29 [sp] = 7ffff524
R30 [s8] = 0
R31 [ra] = 400018

[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] 20100000 addi \$16, \$0, 0 ; 12: addi \$s0, \$zero, 0 # \$s0 (f) = 0
[00400028] 20110001 addi \$17, \$0, 1 ; 13: addi \$s1, \$zero, 1 # \$s1 (g) = 1
[0040002c] 20120002 addi \$18, \$0, 2 ; 14: addi \$s2, \$zero, 2 # \$s2 (h) = 2
[00400030] 20130000 addi \$19, \$0, 0 ; 15: addi \$s3, \$zero, 0 # \$s3 (i) = 0
[00400034] 20140000 addi \$20, \$0, 0 ; 16: addi \$s4, \$zero, 0 # \$s4 (j) = 0
[00400038] 16740003 bne \$19, \$20, 12 [Else-0x00400038] ; 18: add \$s0, \$s1, \$s2 # f = g + h (skipped if i != j)
[0040003c] 02328020 add \$16, \$17, \$18 ; 19: j Exit # go to Exit
[00400040] 08100012 j 0x00400048 [Exit] ; 22: sub \$s0, \$s1, \$s2 # f = g - h (skipped if i = j)
[00400044] 02328022 sub \$16, \$17, \$18 ; 25: li \$v0, 10 # 系統呼叫編號 10 (Exit)
[00400048] 3402000a ori \$2, \$0, 10 ; 26: syscall
[0040004c] 0000000c syscall ; 26: syscall

User Text Segment [00400000]..[00440000]

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 __m1_
[800001a0] 34020004 ori \$2, \$0, 4 ; 103: syscall
[800001a4] 3c049000 lui \$4, -28672 [__m1_] ; 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 __exc0(\$a0)
[800001c0] 3344003c andi \$4, \$26, 60
[800001c4] 3c019000 lui \$1, -28672
[800001c8] 00240021 addu \$1, \$1, \$4

See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.

初始化參數 f~j 對應 s0~s4，If else if 用 bne else 來寫，因為 i=j 成立，程式執行 f=g+h 的邏輯。

由於程式在執行完 THEN 區塊後直接跳轉到 Exit，因此 Else 標籤下的指令會被跳過，不會執行。

最後變數 f 儲存在暫存器 s0 中，最終值為 3。

- (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)

```
ASM while.asm
1 #####
2 # MIPS While 範例
3 #
4 # while (save[i] == k)
5 #     i += 1;
6 #####
7
8 .data
9 save: .word 7, 7, 7, 8, 9 # 陣列 save 的數據 (假設 save[0]=7, save[1]=7, ...)
10
11 .text
12 .globl main
13
14 main:
15     addi $s3, $zero, 0 # i = 0 (從索引 0 開始)
16     addi $s5, $zero, 7 # k = 7 (設定要尋找的值為 7)
17
18     la $s6, save        # $s6 = 載入陣列 save 的基址 (記憶體地址)
19
20 Loop:
21     sll $t1, $s3, 2 # Temp reg $t1 = i * 4
22     add $t1, $t1, $s6 # $t1 = address of save[i]
23     lw $t0, 0($t1) # Temp reg $t0 = save[i]
24     bne $t0, $s5, Exit # go to Exit if save[i] != k
25     addi $s3, $s3, 1 # i = i + 1
26     j Loop # go to Loop
27
28 Exit:
29     li $v0, 10          # 系統呼叫編號 10 (Exit)
30     syscall
31
```

QtSpim

File Simulator Registers Text Segment Data Segment Window Help

FP Regs Int Regs [16] Data Text

Int Regs [16]

PC = 40004c
EPC = 0
Cause = 0
BadVAddr = 0
Status = 3000ff10
HI = 0
LO = 0
R0 [r0] = 0
R1 [at] = 0
R2 [v0] = a
R3 [v1] = 0
R4 [a0] = 1
R5 [a1] = 7ffff530
R6 [a2] = 7ffff538
R7 [a3] = 0
R8 [t0] = 8
R9 [t1] = 1001000c
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] = 3
R20 [s4] = 0
R21 [s5] = 7
R22 [s6] = 10010000
R23 [s7] = 0
R24 [t8] = 0
R25 [t9] = 0
R26 [k0] = 0
R27 [k1] = 0
R28 [gp] = 10008000
R29 [sp] = 7ffff52c
R30 [s8] = 0
R31 [ra] = 400018

User Text Segment [00400000]..[00440000]

[00400000] 8fa40000 lw \$4, 0(\$29) ; 183: lw \$a0 (\$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] 20130000 addi \$19, \$0, 0 ; 15: addi \$s3, \$zero, 0 # i = 0 (從索引 0 開始)
[00400028] 20150007 addi \$21, \$0, 7 ; 16: addi \$s5, \$zero, 7 # k = 7 (設定要尋找的值為 7)
[0040002c] 3c161001 lui \$22, 4097 [save] ; 18: la \$s6, save # \$s6 = 載入陣列 save 的基址 (記憶體地址)
[00400030] 00134880 sll \$9, \$19, 2 ; 21: sll \$t1, \$s3, 2 # Temp reg \$t1 = i * 4
[00400034] 01364820 add \$9, \$9, \$22 ; 22: add \$t1, \$t1, \$s6 # \$t1 = address of save[i]
[00400038] 8d280000 lw \$8, 0(\$9) ; 23: lw \$t0, 0(\$t1) # Temp reg \$t0 = save[i]
[0040003c] 15150003 bne \$8, \$21, 12 [Exit-0x0040003c] ; 25: addi \$s3, \$s3, 1 # i = i + 1
[00400040] 22730001 addi \$19, \$19, 1 ; 26: j Loop # go to Loop
[00400044] 0810000c j 0x00400030 [Loop] ; 29: li \$v0, 10 # 系統呼叫編號 10 (Exit)
[00400048] 3402000a ori \$2, \$0, 10 ; 30: syscall
[0040004c] 0000000c syscall ; 30: 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
[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 __m1
[800001a0] 34020004 ori \$2, \$0, 4 ; 103: syscall
[800001a4] 3c049000 lui \$4, -28672 [__m1] ; 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
[800001c4] 3c019000 lui \$1, -28672

See the file README for a full copyright notice.
QtSPIM is linked to the Qt library, which is distributed under the GNU Lesser General Public License version 3 and version 2.1.

設定 i 為 0

設定 k 為 7

s6 是 save 陣列

s3 是 i

s5 是 k

save 陣列 [7, 7, 7, 8, 9]。

程式從 i=0 開始迴圈。每輪迭代，程式計算 save[i] 的地址 \$t1 並載入值到 \$t0。

當 i=0, 1, 2 時，bne 條件不成立，PC 留在迴圈中，s3 遞增。

當 i=3 時，t0 載入值 8。此時 t0 != s5 (即 8 != 7)，bne \$t0, \$s5, Exit 條件成立，PC 立即跳轉到 Exit 標籤，迴圈終止。

最後，暫存器 s3 (i) 停留在 3，即連續相等元素的個數。