JOINT INSTITUTE
交大密西根学院

Ve370 Introduction to Computer Organization

Homework 3

1. Exercise 2.21.4. (10 points)
2. Exercise 2.21.5. (10 points)
3. Exercise 2.21.6. (10 points)

The following three problems in this Exercise refer to this function, written in MIPS assembly following the calling conventions from Figure 2.14:

a.

```
f: add   $v0,$a1,$a0
   bnez  $a2,L
   sub   $v0,$a0,$a1
L: jr    $v0
```

b.

```
f: add   $a2,$a3,$a2
   slt   $a2,$a2,$a0
   move  $v0,$a1
   beqz  $a2, L
   jr    $ra
L: move  $a0,$a1
   jal   g       ; Tail call
```

2.21.4 [10] <2.8> This code contains a mistake that violates the MIPS calling convention. What is this mistake and how should it be fixed?

2.21.5 [10] <2.8> What is the C equivalent of this code? Assume that the function's arguments are named `a`, `b`, `c`, etc. in the C version of the function.

2.21.6 [10] <2.8> At the point where this function is called register `$a0`, `$a1`, `$a2`, and `$a3` have values 1, 100, 1000, and 30, respectively. What is the value returned by this function? If another function `g` is called from `f`, assume that the value returned from `g` is always 500.

Name	Register number	Usage	Preserved on call?
$zero	0	The constant value 0	n.a.
$v0–$v1	2–3	Values for results and expression evaluation	no
$a0–$a3	4–7	Arguments	no
$t0–$t7	8–15	Temporaries	no
$s0–$s7	16–23	Saved	yes
$t8–$t9	24–25	More temporaries	no
$gp	28	Global pointer	yes
$sp	29	Stack pointer	yes
$fp	30	Frame pointer	yes
$ra	31	Return address	yes

FIGURE 2.14 MIPS register conventions. Register 1, called $at, is reserved for the assembler (see Section 2.12), and registers 26–27, called $k0–$k1, are reserved for the operating system. This information is also found in Column 2 of the MIPS Reference Data Card at the front of this book.

4. Exercise 2.31.1. (5 points)
5. Exercise 2.31.2. (5 points)
6. Exercise 2.31.3. (5 points)

Exercise 2.31

The table below contains the link-level details of two different procedures. In this exercise, you will be taking the place of the linker.

a.

Procedure A					Procedure B				
Text Segment	Address	Instruction			Text Segment	Address	Instruction		
	0	`lbu $a0, 0($gp)`				0	`sw $a1, 0($gp)`		
	4	`jal 0`				4	`jal 0`		
Data Segment	0	(X)			Data Segment	0	(Y)		
	...	...				...	...		
Relocation Info	Address	Instruction Type	Dependency		Relocation Info	Address	Instruction Type	Dependency	
	0	`lbu`	X			0	`sw`	Y	
	4	`jal`	B			4	`jal`	A	
Symbol Table	Address	Symbol			Symbol Table	Address	Symbol		
	–	X				–	Y		
	–	B				–	A		

b.

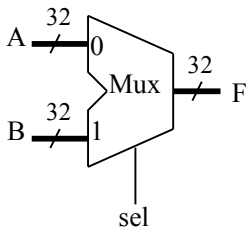
Procedure A					Procedure B				
Text Segment	Address	Instruction			Text Segment	Address	Instruction		
	0	`lui $at, 0`				0	`sw $a0, 0($gp)`		
	4	`ori $a0, $at, 0`				4	`jmp 0`		
	...	...				...	...		
	0x84	`jr $ra`				0x180	`jal 0`		
	...	...				...	...		
Data Segment	0	(X)			Data Segment	0	(Y)		
	...	...				...	...		
Relocation Info	Address	Instruction Type	Dependency		Relocation Info	Address	Instruction Type	Dependency	
	0	`lui`	X			0	`sw`	Y	
	4	`ori`	X			4	`jmp`	FOO	
						0x180	`jal`	A	
Symbol Table	Address	Symbol			Symbol Table	Address	Symbol		
	–	X				–	Y		
						0x180	FOO		
						–	A		

2.31.1 [5] <2.12> Link the object files above to form the executable file header. Assume that Procedure A has a text size of 0x140 and data size of 0x40 and Procedure B has a text size of 0x300 and data size of 0x50. Also assume the memory allocation strategy as shown in Figure 2.13.

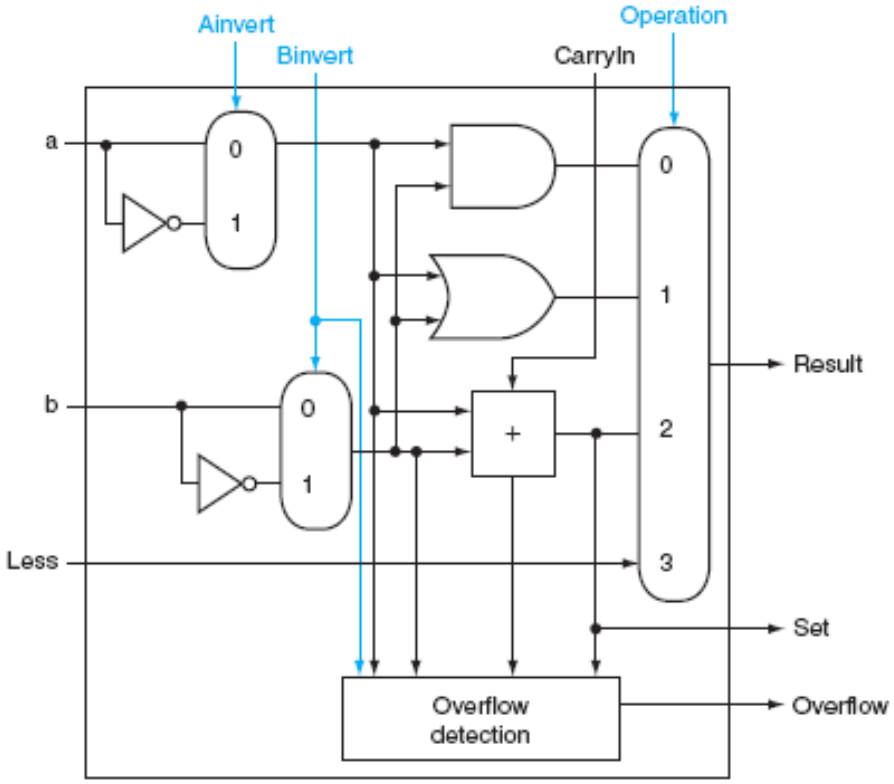
2.31.2 [5] <2.12> What limitations, if any, are there on the size of an executable?

2.31.3 [5] <2.12> Given your understanding of the limitations of branch and jump instructions, why might an assembler have problems directly implementing branch and jump instructions an object file?

7. Describe a 32-bit 2-to-1 MUX in Verilog. Simulate your Verilog module. (10 points)



8. Model the following 1-bit ALU with Verilog. Simulate your Verilog module. Note: the Overflow Detection circuit doesn't have to be included. (20 points)



9. Model a 32 x 32-bit register file shown as following diagram using Verilog. Simulate your Verilog module with Xilinx ISE. Assume the device is not clock triggered. (25 points)