Homework 5
yxu66

2.24 [5] <§2.7> Suppose the program counter (PC) is set to 0x2000 0000. Is it possible to use the jump (j) MIPS assembly instruction to set the PC to the address as 0x4000 0000? Is it possible to use the branch-on-equal (beq) MIPS assembly instruction to set the PC to this same address?

For the jump MIPS assembly instruction, we can't do that. Because 0x4000 0000 - 0x2000 0000 = 0x2000 0000 = 536870912
536870912 / 4 = 134217728 < 2^28= 268435456
For the beq MIPS assembly instruction, we can't do that. Because 0x4000 0000 - 0x2000 0000 = 0x2000 0000 = 536870912
536870912 / 4 = 134217728 > 2^18 = 262144

2.40 [5] <§§2.6, 2.10> If the current value of the PC is 0x00000000, can you use a single jump instruction to get to the PC address as shown in Exercise 2.39?

Creates the 32-bit constant 0010 0000 0000 0001 0100 1001 0010 0100two and stores that value to register $t1 means need to store 0x2001 4924 into $t1.
Current address: 0x0000 0000
0010 0000 0000 0001 0100 1001 0010 0100two = 0x2001 4924(hex)
0x2001 4924 - 0x0000 0000 = 0x2001 4924 = 536955172
536955172 / 4 = 134238793 < 2^28 = 268435456

So we can use a single j instruction to get the PC address.

2.41 [5] <§§2.6, 2.10> If the current value of the PC is 0x00000600, can you use a single branch instruction to get to the PC address as shown in Exercise 2.39?

current address: 0x00000600
0010 0000 0000 0001 0100 1001 0010 0100two = 0x2001 4924(hex)
0x2001 4924 - 0x0000 0600 = 536953636
536953636 / 4 = 134238409 > 2^18 = 262144
So we cannot use branch instruction to get the PC address.

2.42 [5] <§§2.6, 2.10> If the current value of the PC is 0x1FFFf000, can you use a single branch instruction to get to the PC address as shown in Exercise 2.39?

Current address: 0x1FFFf000
0010 0000 0000 0001 0100 1001 0010 0100two = 0x2001 4924(hex)
0x2001 4924 - 0x1FFFf000 = 88356
88356 / 4 = 22089 < 2^18 = 262144
So we can use a single branch instruction to get the PC address

This diagram shows the object files for two pairs of functions. Procedure A has a

text size of 0x140 and a data size of of 0x40. Procedure B has a text size of 0x300 and a data size of 0x50. Show the executables created by the linker for these two pairs of procedures if the linker uses the memory layout shown in the MIPS Green sheet.

1) From the information, we know:

Text of A:	0x0040 0000	lbu $a0, 0($gp)	#X
	0x0040 0004	jal 0	#B
	…		
Text of B:	0x0040 0140	sw $a1, 0($gp)	#Y
	0x0040 0144	jal 0	#A
	…		
Data of A:	0x1000 0000	X	
	…		
	0x1000 0040	Y	
	…		

offset = 0x1000 0000 - 0x1000 8000 = -0x8000 = 0xf8000
offset = 0x1000 0040 - 0x1000 8000 = -0x7fc0 = 0xf8040
so excitable file:

Excutable file header

Text segment:	Text size:	0x440
	Data size:	0x90
	Address	**Instruction**
	0x0040 0000	lbu $a0, 0xf8000 ($gp)
	0x0040 0004	jal 0x0040 0140
	…	
	0x0040 0140	sw $a1, 0xf8040($gp)
	0x0040 0144	jal 0x0040 0000
	…	
Data segment:	**Address**	
	0x1000 0000	X
	…	
	0x1000 0040	Y
	…	

2) The same as above:

Text of A:	0x0040 0000	lui $at, 0	#X
	0x0040 0004	ori $a0, $at, 0	#X
	…		
	0x0040 0084	jr $ra	
	…		
Text of B:	0x0040 0140	sw $a0, 0($gp)	#Y
	0x0040 0144	jmp 0	#F00
	…		
	0x0040 0180	jal 0	#A
Data of A:	0x1000 0000	X	
	…		
	0x1000 0040	Y	
	…		

Excutable file header

Text segment:	Text size:	0x440
	Data size:	0x90
	Address	**Instruction**
	0x0040 0000	lui $at, 0x1000 0000
	0x0040 0004	ori $a0, $at, 0x1000 0000
	…	
	0x0040 0084	jr $ra
	…	
	0x0040 0140	sw $a0, 0xf8040($gp)
	0x0040 0144	jmp 0x0040 0180
	…	
	0x0040 0180	jal 0x0040 0000
	…	
Data segment:	**Address**	
	0x1000 0000	X
	…	
	0x1000 0040	Y
	…	

offset = 0x1000 0040 - 0x1000 8000 = -0x7fc0 = 0xf8040