Solution

Problem 1: (18points)

1 **4**

2 [1] **1** [2] **1** [3] **2**

3 **3**
exit status=2
exit status=3
exit status=4
... (6 possible)

4 **1**

exit status=2 or exit status=3 or exit status=4

Problem 2: (24 points)

1 [1] **8** [2] **2** [3] **2**

2 0100 0101 0100 ~ 0100 0101 0111 (0x454 ~ 0x457)

3 [1] **H** [2] **0x3086**
[3] **M** [4] **--**
[5] **M** [6] **--**
[7] **H** [8] **0x4F60**

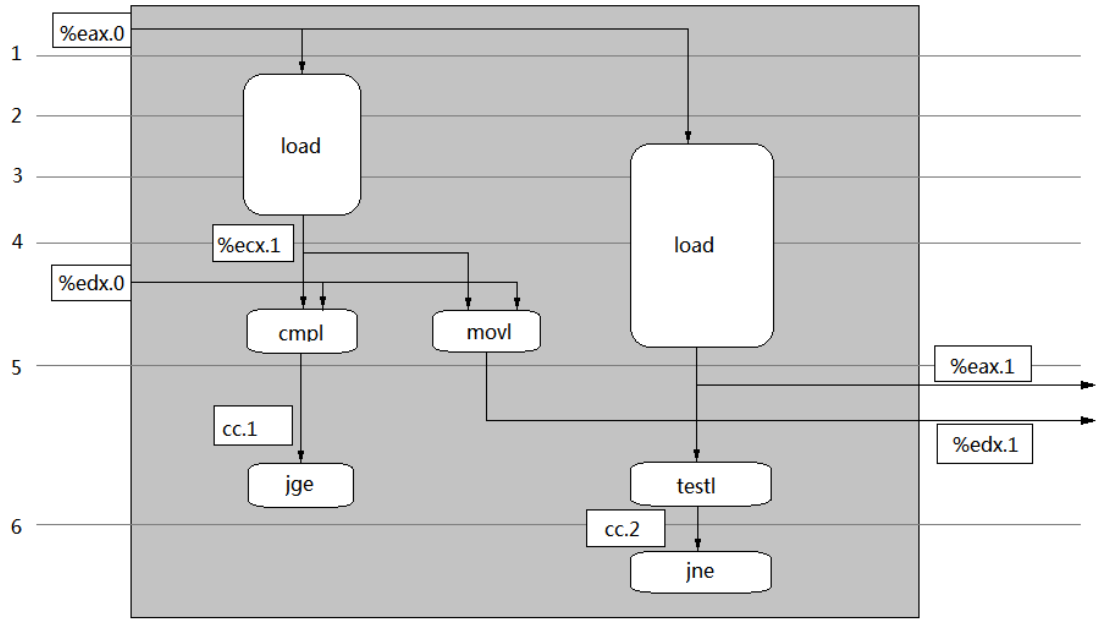
4 1) **16**
2) **1/2**
3) **32**
4) **1**
5)
[1] **02** [2] **1** [3] **02 00 03 00**
[4] **03** [5] **1** [6] **03 00 04 00**

Problem 3: (16 points)

1

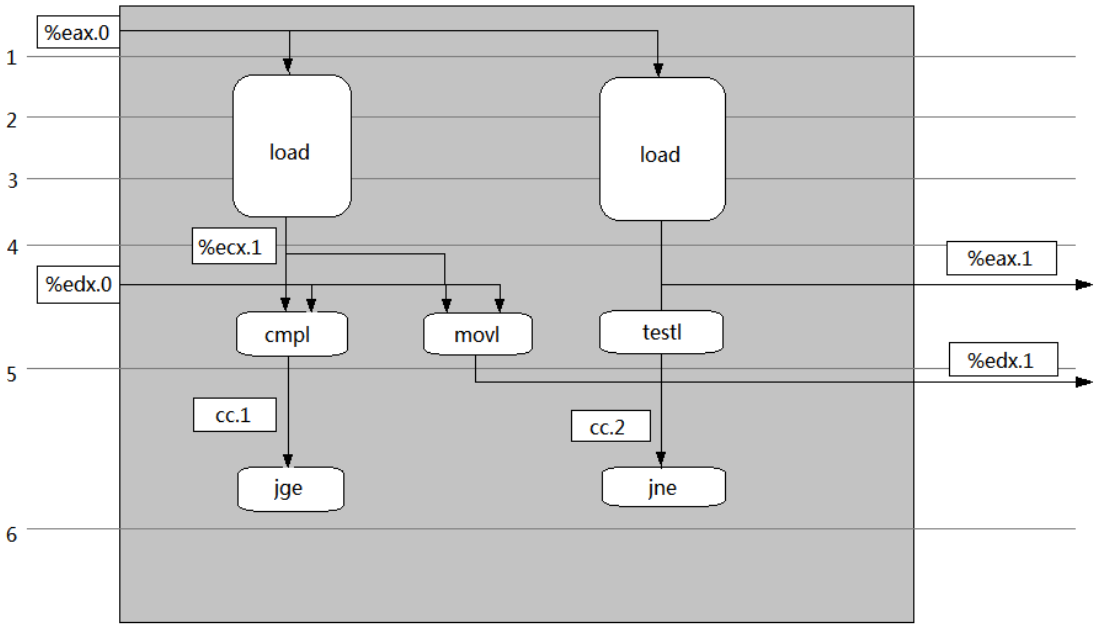
```
   .L1
1. movl     (%eax),%ecx        load   (%eax.0) → %ecx.1
2. cmpl     %ecx,%edx          cmpl   %ecx.1, %edx.0 → %cc.1
3. jge      .L2                jge-not-taken %cc.1
4. movl     %ecx,%edx          movl   %ecx.1, %edx.1
   .L2
5. movl     4(%eax),%eax       load   4(%eax.0) → %eax.1
6. testl    %eax,%eax          testl  %eax.1, %eax.1 → %cc.2
7. jne      .L1                jne-taken    %cc.2
```

2



CPE = 4.0

3



CPE = 3.0

4 **The bottleneck of the program is the data dependence on register %eax between two adjacent iterations.**

Problem 4: (24 points)

1 1) **7**

2) **8**

3) No. The predicted PC can’t be calculated until the fetch stage is finished for the jmp instruction.

2 [1]

icode:ifun←"M" _"1" [PC]

rA:rB←"M" _"1" [PC+1]

valC←"M" _"4" [PC+2]

valP←PC+6

[2]

valA←R[rA]

valB←R[rB]

[3]

valM←"M" _"4" [valE]

"M" _"4" [valE]←valA

[4]

R[rA]←valM

3

[1]	**M**	[2]	**E**	[3]	**E**	[4]	**--**
[5]	**done**	[6]	**E**	[7]	**W**	[8]	**M**
[9]	**11**	[10]	**--**	[11]	**0x44F**		
[12]	**W**	[13]	**D**	[14]	**M**		
[15]	**6**	[16]	**0x44F**				

4 **18**

F	F								1
D	D	F	F						2
E	E	D	D	F	F				3
E	E	D	E	D	F	F			4
M	M	E	E	D	D	F	F		5
W	W	E	M	E	D	D	F	F	6
		E	W	E	D	D	F	F	7
		M		M	D	E	D	F	8
		W		M	D	E	E	D	9
				M	D	E	E	D	10
				W	E	M	M	D	11
					E	W	W	D	12
					M			E	13
					W			E	14
								M	15
								M	16
								M	17
								W	18