1. Use the Booth algorithm to calculate (-4) x (6)
   Note that you have to write the process step by step. (10 pts)

2. Assume that a 4-way set associative cache has 1024 entries. Each entry has four-way data blocks, each 4 words long (1 word=4 bytes). Also, each data block has three status bits, V (valid), D(dirty), and R(reference). This cache is byte-addressable and the length of the address is 32-bit.
   (a) Calculate the total size of this cache in bits. (10 pts)
   (b) Explain the function of each status bit. (10 pts)

3. Fig. 1 shows the pipelined datapath of the MIPS processor. There is a code sequence:

```
lw   $1, 40($6)
add  $3, $2, $1
sw   $3, 40($6)
add  $4, $3, $zero
lw   $1, 44($6)
add  $3, $2, $1
sw   $3, 44($6)
add  $5, $3, $4
```

   Answer the following questions.
   (a) Identify where will the data hazards happen. (10 pts)
   (b) If we implement the data forwarding path in Fig. 1, how many stall cycles will this code sequence have? (10 pts)
   (c) Continued with (b). If you are allowed to rename the register names, is it possible to schedule (reorder) these instructions to reduce the stall cycles from the data hazards but still maintaining correctness? Show how to do it. (10 pts)

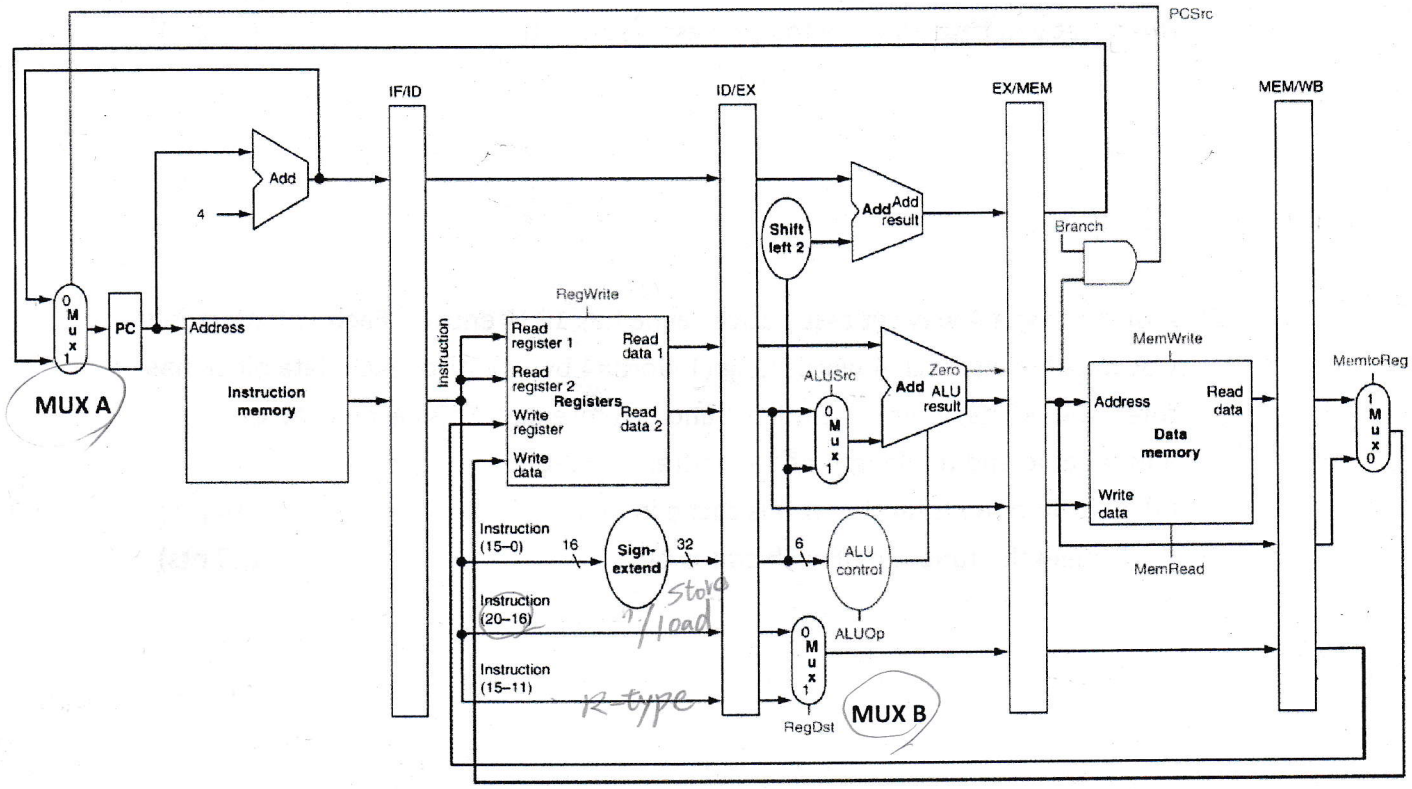

Fig. 1

4. Check the multiplexers "MUX A" and "MUX B" in Fig 1, and answer the following questions. (40 pts, 5 pts each)
   (a) At the first stage (IF), why we need "MUX A" before the PC register?
   (b) How to modify "MUX A" such that it can support both of "BEQ" and "BNE" instructions? Explain your answer in details.
   (c) At the third stage (EXE), why we need "MUX B"?
   (d) Can we move "MUX B" to the second stage (ID) such that it can work earlier?
   (e) Take a code sequence for example to explain why the datapath of Fig. 1 will cause control hazards.
   (f) The method "Predict-Taken" has been considered a better way to reduce control hazards. Explain how to implement the "Predict-Taken" path in Fig. 1.
   (g)&(h) Give two instructions as follows, fill their control signals in the table.

Control signals	RegWrite	RegDst	MemWrite	MemRead	MemtoReg
`lw $t1, 40($t2)`	1	0	0	1	1
`add $t3, $t2, $t1`	1	1	0	0	0