Computer Organization and Structure

Homework #5
Due: 2007/1/9

1. If the time for an ALU operation can be shortened by 25% (compared to the following table);

Instruction class	Instruction fetch	Register read	ALU operation	Data access	Register write	Total time
Load word (`lw`)	200 ps	100 ps	200 ps	200 ps	100 ps	800 ps
Store word (`sw`)	200 ps	100 ps	200 ps	200 ps		700 ps
R-format (`add`, `sub`, `and`, `or`, `slt`)	200 ps	100 ps	200 ps		100 ps	600 ps
Branch (`beq`)	200 ps	100 ps	200 ps			500 ps

   a. Will it affect the speedup obtained from pipelining? If yes, by how much? Otherwise, why?
   b. What if the ALU operation now takes 25% more time?

2. Identify all of the data dependencies in the following code. Which dependencies are data hazards that will be resolved via forwarding? Which dependencies are data hazards are will cause a stall?

```
add   $3, $4, $2
sub   $5, $3, $1
lw    $6, 200($3)
add   $7, $3, $6
```

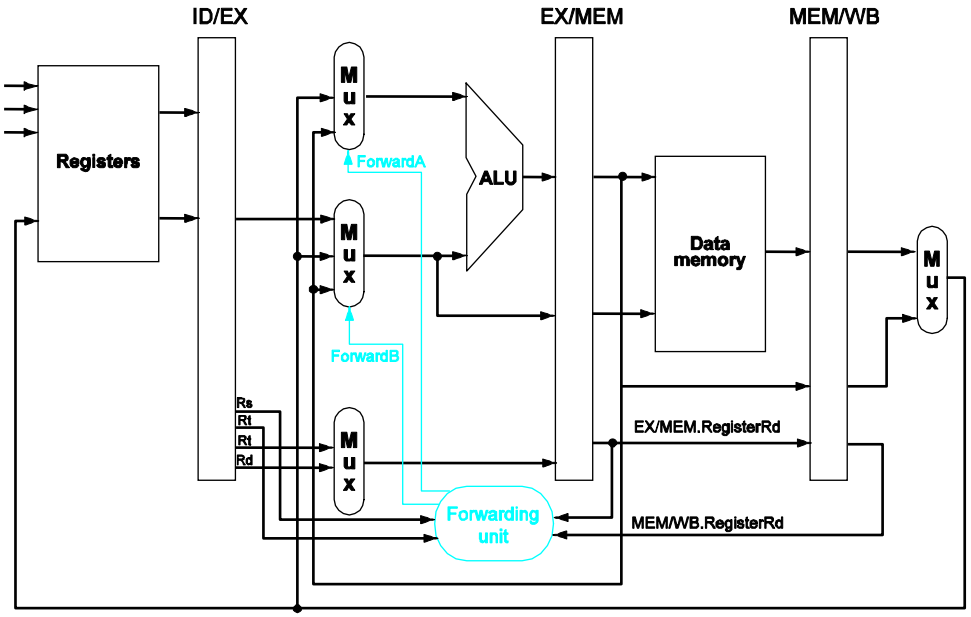
3. The following piece of code is executed using the pipeline shown in the following figure:

```
lw    $5, 40($2)
add   $6, $3, $2
or    $7, $2, $1
and   $8, $4, $3
sub   $9, $2, $1
```

   At cycle 5, right before the instructions are executed, the processor state is as follows:

   a. The PC has the value 100_{ten}, the address of the `sub_instruction`.
   b. Every register has the initial value 10_{ten} plus the register number (e.g., register `$8` has the initial value 18_{ten}).
   c. Every memory word accessed as data has the initial value 1000_{ten} plus the byte address of the word (e.g., Memory[8] has the initial value 1008_{ten}).

   Determine the value of every field in the four pipeline registers in cycle 5.



4. The performance can be *maximized* on the pipelined datapath by using forwarding and stalls on a use following a load. Rewrite the following code to *minimize* performance on this datapath – that is, reorder the instructions so that this sequence takes the *most* clock cycles to execute while stall obtaining the same result.

```
lw    $2, 100($6)
lw    $3, 200($7)
add   $4, $2, $2
add   $6, $3, $5
sub   $8, $4, $6
lw    $7, 300($8)
beq   $7, $8, Loop
```