Homework#2 - Solution

1. [15 points] This problem covers 4-bit binary multiplication. Fill in the table for the Product, Multplier and Multiplicand for each step. You need to provide the DESCRIPTION of the step being performed (shift left, shift right, add, no add). The value of M (Multiplicand) is 1011, Q (Multiplier) is initially 1010.
   (M과 Q에 대한 shift left, shift right, add, no add 등은 각각 분리된 step으로 구분할 것)

Product	Multiplicand	Multiplier	Description	Step
0000 0000	0000 1011	1010	Initial Values	Step 0
0000 0000	0000 1011	1010	0 => No add	Step 1
0000 0000	0001 0110	1010	Shift left M	Step 2
0000 0000	0001 0110	0101	Shit right Q	Step 3
0001 0110	0001 0110	0101	1 => Add M to Product	Step 4
0001 0110	0010 1100	0101	Shift left M	Step 5
0001 0110	0010 1100	0010	Shift right Q	Step 6
0001 0110	0010 1100	0010	0 => No add	Step 7
0001 0110	0101 1000	0010	Shift left M	Step 8
0001 0110	0101 1000	0001	Shift right Q	Step 9
0110 1110	0101 1000	0001	1 => Add M to Product	Step 10
0110 1110	1011 0000	0001	Shift left M	Step 11
0110 1110	1011 0000	0000	Shift right Q	Step 12
0110 1110	1011 0000	0000	0 => No add	Step 13
0110 1110	0110 0000	0000	Shift left M	Step 14
0110 1110	0110 0000	0000	Shift Right Q	Step 15

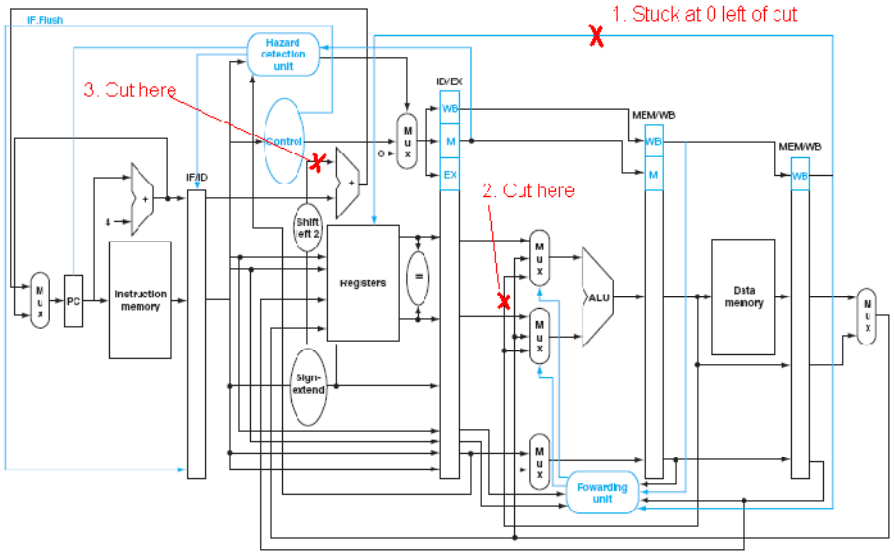
➔ Step 12까지 기술하면 정답으로 인정

2. [10 points] Assuming single precision IEEE 754 format, what decimal number is represent by this word:
   1 01111101 00100000000000000000000

   The decimal number
   $= (-1)^1 \times 2^{125-127} \times 1.001_2$
   $= (-1) \times 0.25 \times 1.125 = -0.28125$

3. [10 points] The floating-point format to be used in this problem is an 8-bit IEEE 754 normalized format with 1 sign bit, 4 exponent bits, and 3 faction bits. It is identical to the 32-bit and 64-bit formats in terms of the meaning of fields and special encodings. The bit fields in a number are (sign, exponent, faction). Assume that we use unbiased *rounding to the nearest even* specified in the IEEE floating point standard.
   Encode 0.0011011_2 the 8-bit IEEE format.
   Normalizing this number yields:
   $1.1011_2 \times 2^{-3}$
   $= (+1) \times 1.1011_2 \times 2^{(4-7)}$
   = 0 0100 110 in the 8-bit IEEE 754
   (after applying unbiased rounding to the nearest even)

4. [15 points] For the MIPS datapath shown below, several lines are marked with "X". For each one:
   A. Describe in words the negative consequence of cutting this line relative to the working, unmodified processor.
   B. Provide a snippet of code that will fail



(1) Cannot write to register file. This means that R-type and any instruction with write back to register file will fail. An example of code snippet that would fail is:

add $s1, $s2, $s3

(2) Forwarding of the first operand fails. An example of code snippet that would fail is:

add $s1, $t0, $t1
add $s1, $s1, $s1

(3) Jumping to a branch target does not work. Example of code that fails:

addi $s1, $zero, 2
addi $s2, $zero, 2
beq $s1, $s2, exit

5. [20 points] Consider the following code:

```
Loop:   lw    $1, 0($2)          # R1 ← M[0+R2]
        addi  $1, $1, 1          # R1 ← R1 + 1
        sw    $1, 0($2)          # M[0+R2] ← R1
        addi  $2, $2, 4          # R2 ← R2 + 4
        sub   $4, $3, $2         # R4 ← R3 + R2
        bne   $4, $0, Loop       # Branch to Loop if R4 != R0
```

This code snippet will be executed on a MIPS pipelined processor with a 5-stage pipeline. Branches are resolved in the decode stage and *do not* have delay slots. All memory accesses take 1 clock cycle. In the following three parts, you will be filling out pipeline diagrams for the above code sequence. Please use

acronyms F, D, X, M and W for the 5 pipeline stages.

A. Fill in the pipeline diagram below for the execution of the above code sequence *without* any forwarding or bypassing hardware but assuming a register read and a write in the same clock cycle "forwards" through the register file.

	Cycle																
instruction	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
`lw $1, 0($2)`	F	D	X	M	W												
`addi $1, $1, 1`		F	D	D	D	X	M	W									
`sw $1, 0($2)`			F	F	F	D	D	D	X	M	W						
`addi $2, $2, 4`						F	F	F	D	X	M	W					
`sub $4, $3, $2`									F	D	D	D	X	M	W		
`bne $4, $0, Loop`										F	F	F	D	D	D	X	M
`lw $1, 0($2)`																F	D

B. Fill in the pipeline diagram below for the execution of the above code sequence with traditional pipeline forwarding:

	Cycle																
instruction	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
`lw $1, 0($2)`	F	D	X	M	W												
`addi $1, $1, 1`		F	D	D	X	M	W										
`sw $1, 0($2)`			F	F	D	X	M	W									
`addi $2, $2, 4`					F	D	X	M	W								
`sub $4, $3, $2`						F	D	X	M	W							
`bne $4, $0, Loop`							F	D	D	X	M	W					
`lw $1, 0($2)`										F	D	X	M	W			

6. [15 points] Structural, data and control hazards typically require a processor pipeline to stall. Listed below are a series of optimization techniques implemented in a compiler or a processor pipeline designed to reduce or eliminate stalls due to these hazards. For each of the following optimization techniques, state which pipeline hazards it addresses and how it addresses it. Some optimization techniques may address more than one hazard, so be sure to include explanations for all addressed hazards.

   A. Branch Prediction

It addresses control hazards by guessing the outcome of a branch instruction and then speculatively executes the instructions on one side of the branch to keep the pipeline moving. Predictions can be made in hardware or in software by the compiler.

   B. Instruction Scheduling

It addresses structural hazards and data hazards. It addresses data hazards by either moving instructions that are not dependent on an instruction, say A, before some instructions that depend on A and thus avoiding the stall that would have occurred otherwise. It addresses structural hazards by making sure instructions that use functional units that have limited number of instances are be scheduled far apart from each other and there is no unnecessary stall due to this. It can be done in hardware (superscalar processor) or statically by the compiler

   C. delay slots

It addresses control hazards. It helps to avoid a stall that would result due branch target identification during the decode stage by scheduling the execution of some other instruction which anyway has to execute irrespective of the branch condition.

7. [15 points] Branch Prediction. Consider the following sequence of actual outcomes for a single static branch. T means the branch is taken. N means the branch is not taken. For this question, assume that this is the only branch in the program.

T T T N T N T T T N

Assume that we try to predict this sequence with a BHT using one-bit counters. The counters in the BHT are initialized to the N state. Which of the branches in this sequence would be mis-predicted? Use the following table for your answer:

Predictor state before prediction (T/N)	Branch outcome (T/N)	Misprediction? (Yes/No)
N	T	Y
T	T	N
T	T	N
T	N	Y
N	T	Y
T	N	Y
N	T	Y
T	T	N
T	T	N
T	N	Y