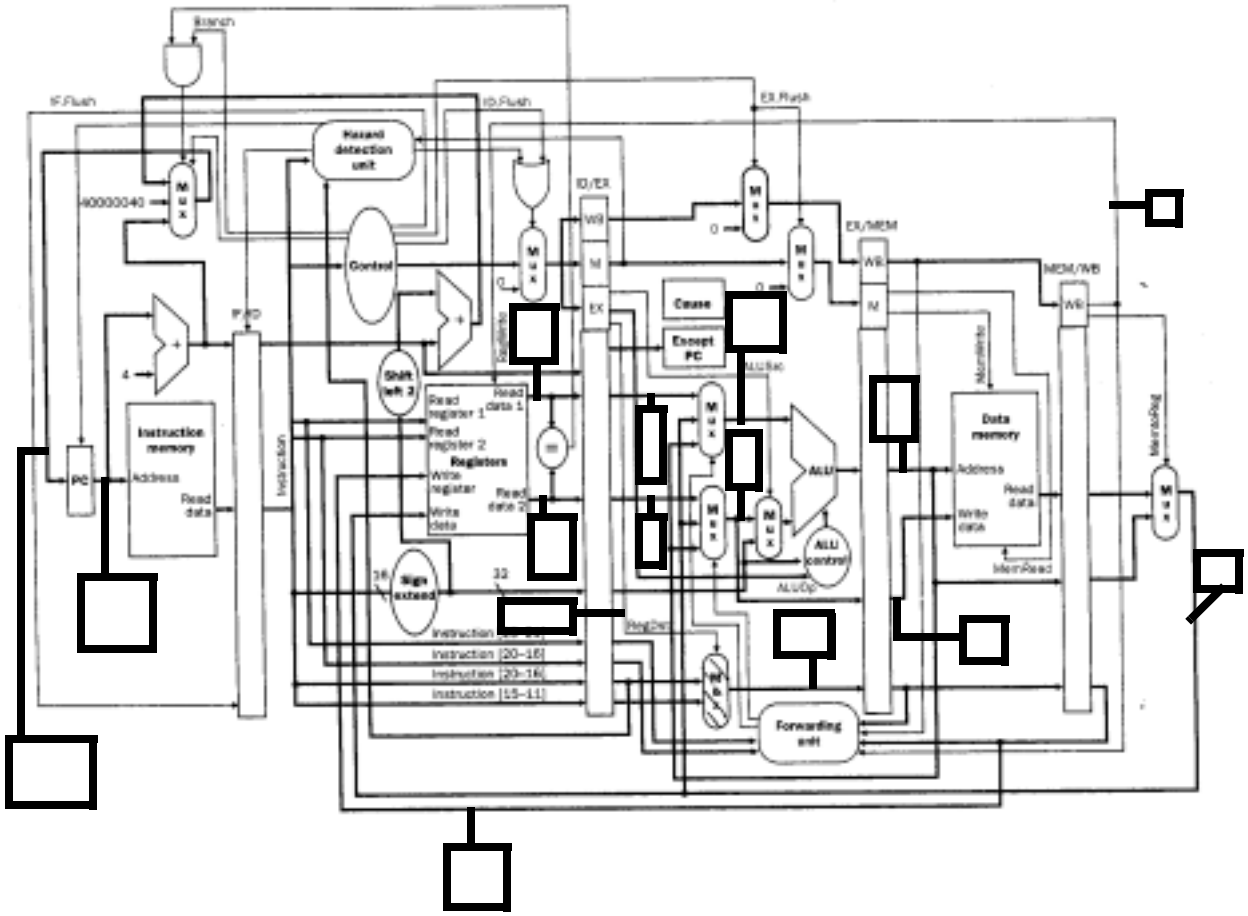
Problem 1: (25 points)

The code below is to be run on the pipelined datapath shown below. Assume the datapath has hardware support for forwarding and the branch operation is performed in the Decode stage. Also a register can be written and read during the same clock cycle.

IF:	ID:	EX:	MEM:	WB:



Show the state of the datapath for the following code when the first ***lw*** instruction is in the ***WB*** stage by filling in the boxes on the datapath with the values for the indicated lines. If you do not know the value in a register, use the notation $X to indicate the contents of register X and M($X) to indicate the contents at memory location X. Assume the branch **IS** taken, but that branch-delay slots are not filled. **Be sure to write the instruction above each stage.**

PC		
A000h	li	$1,8
A004h	li	$2,5
A008h	li	$3,1
A00Ch Loop:	lw	$1, 0($2)
A010h	sub	$3, $3, $2
A014h	add	$4, $3, $1
A018h	bne	$3, $0, Loop
A01Ch	lw	$5, 0($2)

Problem 2: (25 pts)

For the following code, assume that the branch is taken. Draw the multicycle diagram for the first three cycles assuming that **NO** data forwarding takes place. Make sure that your diagram accounts for any control and data hazards by stalling or flushing the pipeline. Assume that all instructions are stalled in the ID stage and flushed in the IF stage (like the MIPS pipeline). If you need more cycles, add them to the end.

```
Loop: lw   $1, 0($2)
      add  $1, $1, $8
      subi $3, $1, 4
      sw   $2, 4($1)
      bne  $3, $0, Loop
      addi $10, $9, 10
```

	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14	c15
lw															
add															
subi															

(b) Redraw the diagram assuming that data can be forwarded to any other stage. Indicate forwarding with an arrow.

	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14	c15
lw															
add															
subi															

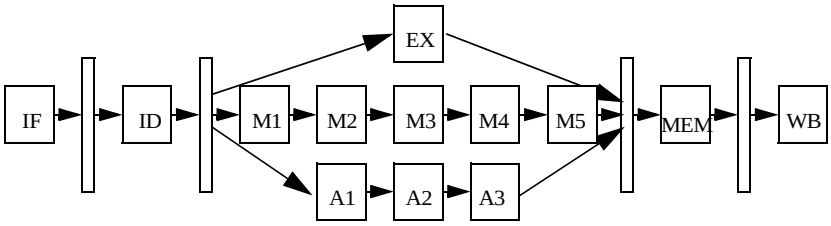
(c) If the branch delay slot was filled by the compiler with an independent instruction, what would happen to the code listed above, as it is written? (Assume the code has been written to fill the delay slot)

(d) If the branch delay slot was filled using dynamic branch prediction, which predicted **correctly** that the branch was **taken**, what instruction would occur after the branch?

Problem 3: (25 points)

A new processor has been designed that executes floating point and integer operations in parallel with latencies as shown in the table below. This pipelined data path performs it's branch decisions in the decode stage with one delay slot after the branch. There is hardware support for data forwarding.

Unit	Latency
Integer Exec	0
Data Memory	1
FP Add	2
FP Mult	4



(a) Insert nop instructions so that the loop executes correctly on the new pipeline. Assign multiple nops as "# nop" to decrease clutter.

```
Loop   ld     $f0, 0($t1)
       multd$f0, $f0, $f2
\
       ld     $f4, 0($t2)
       addd $f0, $f0, $f4
       sd     $f0, 0($t2)
       subi   $t1, $t1, 8
       subi   $t2, $t2, 8
       bne    $t1, $0, Loop
```

(b) Unroll the code once (i.e. two iterations) and schedule to minimize nops. Assume the number of iterations of the loop is an even number. **Show your nops in your scheduled/unrolled code**.

Problem 4: (25 points)

Short Answer - Please underline the key words in your answer.

(a) For an index with a given number of bits, why should gshare perform better than gselect?

(b) You just bought the newest CPU for your machine that has the latest and greatest branch prediction algorithm, which can achieve an accuracy of 97% for the SPEC'2000 benchmark. You program your machine and find out that the branch predictor only achieves a 60% accuracy. Why would this occur? Relate your answer specifically to the assumptions that make branch prediction successful.

(c) What are precise vs. imprecise exceptions? How does the non-superscalar MIPS pipeline maintain precise exceptions?