FACULTY OF ENGINEERING AND ARCHITECTURE **SPRING 2005**
AMERICAN UNIVERSITY OF BEIRUT
EECE 321 – COMPUTER ORGANIZATION

PROF. MANSOUR **MIDTERM 2** MAY 10, 2005

NAME: _____ ID: _____

TIME: 6:00 – 8:00 p.m.

INSTRUCTIONS:

- **CLOSED BOOK/CLOSED NOTES**
- **TIME: 2 HOURS**
- **WRITE YOUR NAME AND ID NUMBER IN THE SPACE PROVIDED ABOVE.**
- **WRITE YOUR ANSWERS ON THE QUESTION SHEET.**
- **THE SCRATCH BOOKLET WILL NOT BE CONSIDERED IN GRADING.**
- **BE AS CLEAR AND AS NEAT AS POSSIBLE WHEN MODIFYING DATAPATHS.**
- **WRITE DOWN ANY ASSUMPTIONS YOU USE IN SOLVING ANY PROBLEM.**

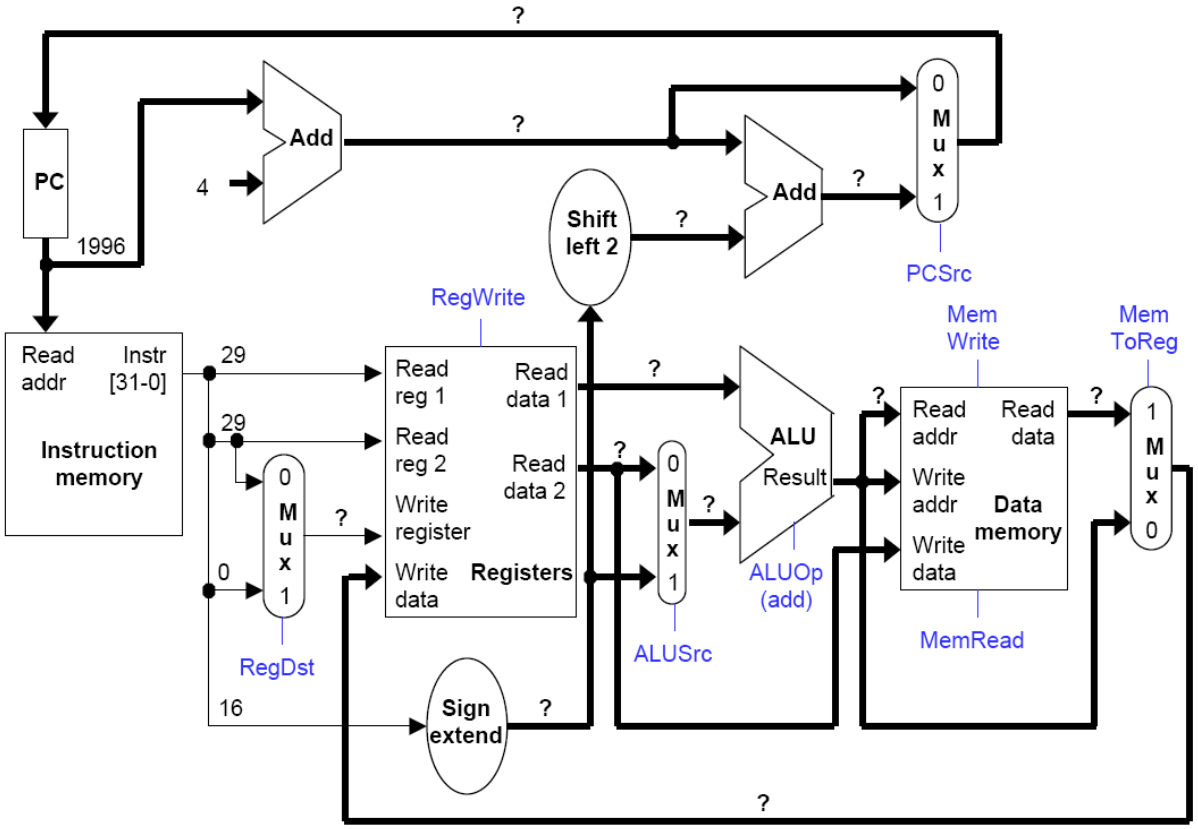
PROBLEM	MAX POINTS	GRADE
1	12	
2	16	
3	20	
4	14	
5	22	
6	42	
7	16	
8	18	
TOTAL	160	

Problem 1: Single-Cycle Datapath [12 Points]

We want to execute the following immediate addition instruction in the single-cycle datapath:

`addi` $29, $29, 16

The single-cycle datapath diagram below shows the execution of this instruction. Several of the datapath values are filled in already. You are to provide values in decimal for the twelve remaining signals in the diagram, which are marked with a **?** symbol. Assume register $29 initially contains the number 129. If a value cannot be determined, mark it as 'X'.



Problem 2: **Multi-Cycle MIPS [16 Points]**

We would like to add the following instruction to the MIPS multi-cycle datapath shown below:

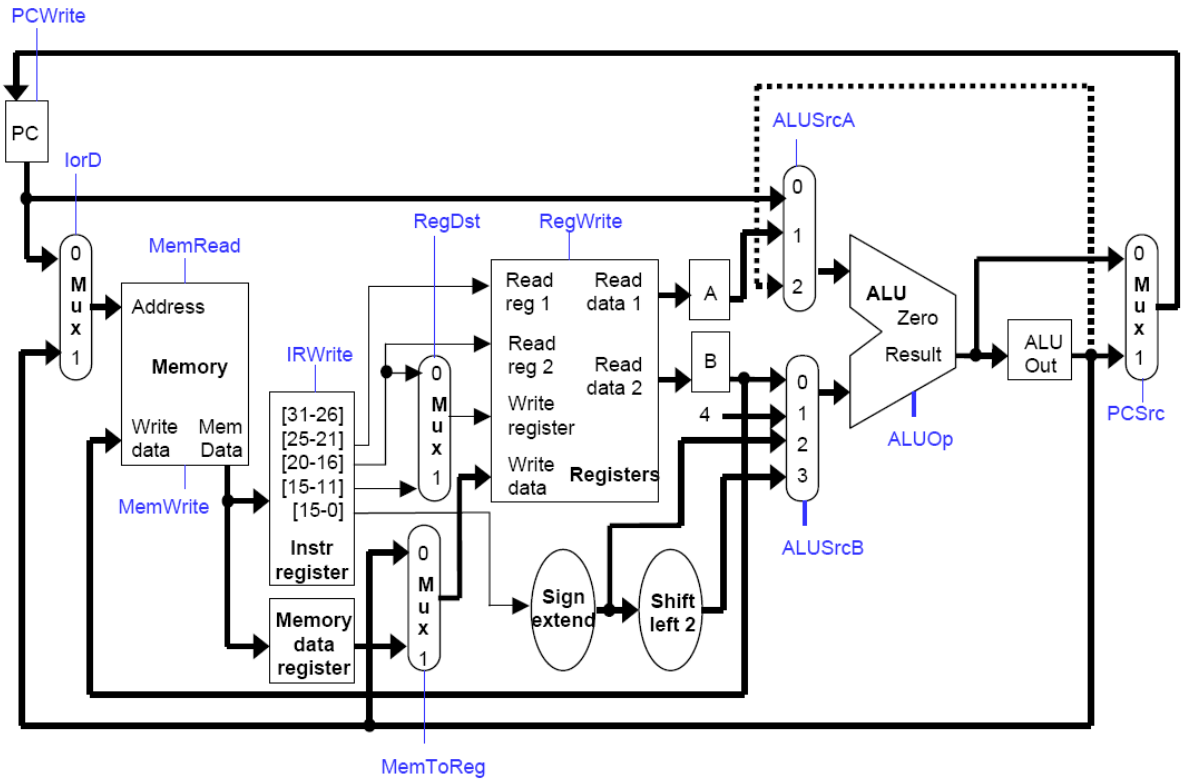
```
lws rd,rs,rt          # rd = Mem[rt + (4 × rs)]
```

For example, if **$a0** contains 1000 and **$a1** contains 10, then "**lws $t0,$a1,$a0**" loads **$t0** with the value at address 1040 (1000 + 4x10).

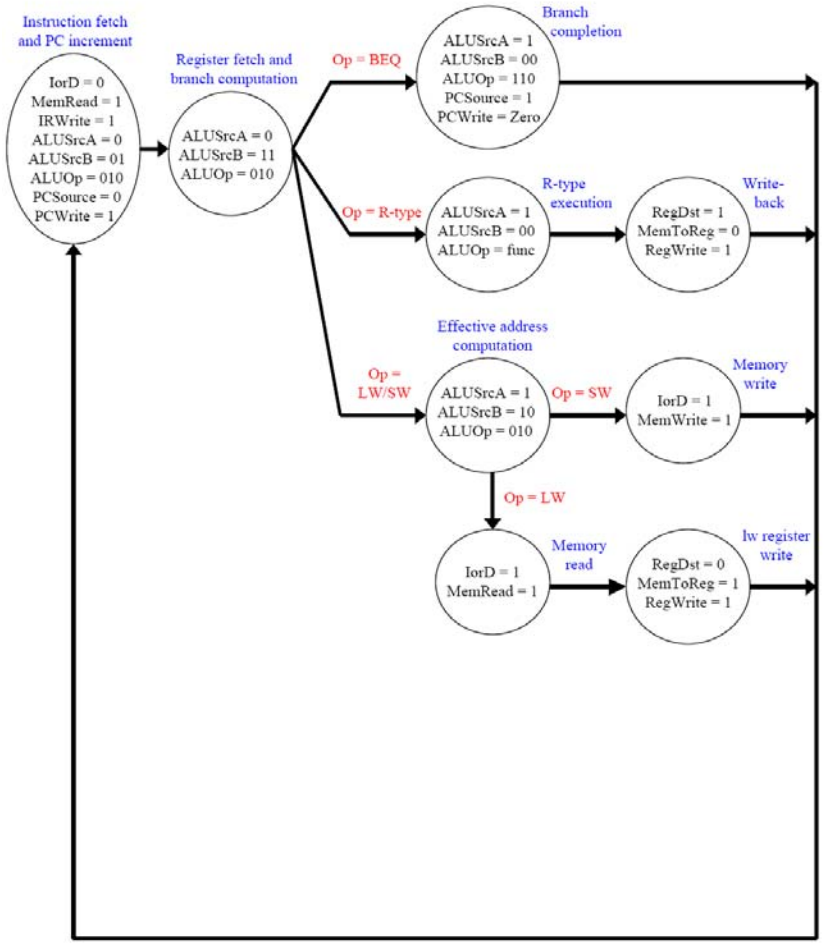
You need to show the correct control signals necessary to implement the **lws** instruction, by (1) completing the finite state machine diagram [**8 Points**], and (2) filling in the microprogramming table shown on the next page [**8 Points**]. Show the control values in either binary or decimal on the FSM diagram. For the microcode, you may need to make up new values for some of the fields shown, and they should be consistent with the FSM.

Assume that ALUOp = 100 performs an integer multiplication, and 010 performs an addition. Finally, here is the instruction format, for your reference (**shamt** and **func** are not used):

Field	op	rs	rt	rd	shamt	func
Bits	31-26	25-21	20-16	15-11	10-6	5-0



(Problem 2 cont'd)

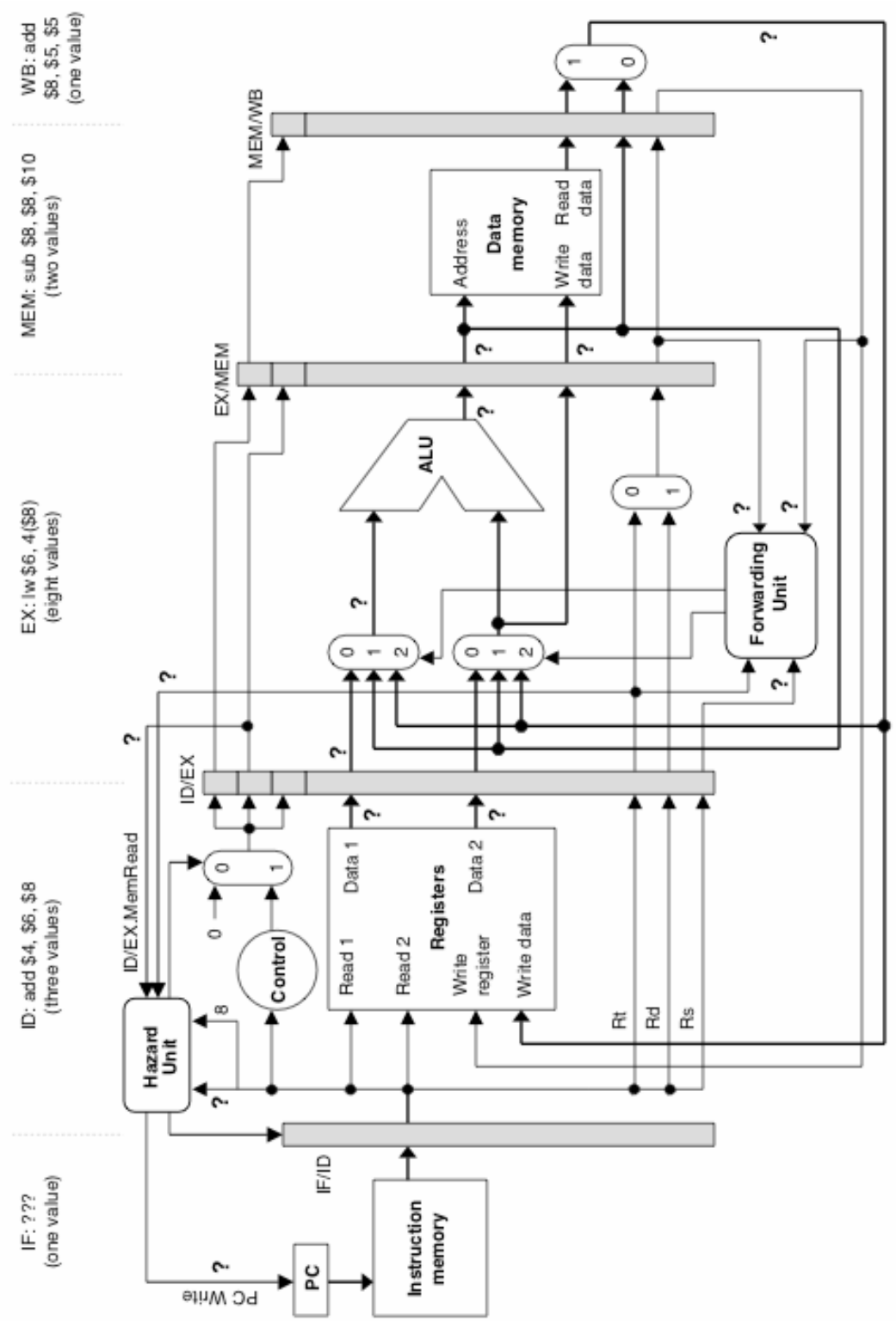


Label	ALU Control	Src1	Src2	Register control	Memory	PCWrite control	Next
Fetch	Add	PC	4		Read PC	ALU	Seq
	Add	PC	Extshift	Read			Dispatch 1
BEQ1	Sub	A	B			ALU-Zero	Fetch
Rtype1	Func	A	B				Seq
				Write ALU			Fetch
Mem1	Add	A	Extend				Dispatch 2
SW2					Write ALU		Fetch
LW2					Read ALU		Seq
				Write MDR			Fetch

Problem 3: **Pipelining and Forwarding** [**20 Points**]

The pipelined datapath below shows the fifth cycle of executing the following program, including values for several of the stages. Fill in the ten remaining values, marked with a **?** symbol, in the EX and MEM stages. Write your answers in decimal directly on the diagram. Assume that registers initially contain their number plus 100: e.g., $2 contains 102, $8 contains 108, etc. Write 'X' for any numbers that cannot be determined.

```
add $8, $5, $5
add $2, $5, $8
sub $3, $8, $4
add $2, $2, $3
```



Problem 4: **Pipelining Performance** [**14 Points**]

IPG	FET	ROT	EXP	REN	WLD	REG	EXE	DET	WRB
1	2	3	4	5	6	7	8	9	10

A designer has proposed modifying the pipelined MIPS datapath to the 10-stage pipeline above so that the resulting processor can run at 500MHz (2ns clock cycle). The correspondences between this and the standard MIPS pipeline are:

- Instructions are fetched in the FET stage.
- Register reading is performed in the REG stage.
- ALU operations *and* memory accesses are both done in the EXE stage.
- Branches are resolved in the DET stage.
- WRB is the writeback stage.

a) How much time is required to execute one million instructions on this processor, assuming there are no dependencies or branches in the code? [**2 Points**]

Solution:

b) *Without forwarding*, how many stall cycles are needed for the following code fragment? [**3 Points**]

```
lw  $t0,0($a0)
add $v1,$t0,$t0
```

Solution:

c) If a branch is mis-predicted, how many instructions would have to be flushed from the pipeline? [**4 Points**]

Solution:

d) Assume that a program executes one million instructions. Of these, 15% are load instructions which stall, and 10% of the instructions are branches. The CPU predicts branches correctly 75% of the time. How much time will it take to execute this program? [**5 Points**]

Solution:

Problem 5: **Cache Computations** [**22 Points**]

The Trashium processor has a 16KB, 4-way set-associative data cache with 32-byte blocks.

a) How many total blocks are in the Level 1 cache? How many sets are there? [**3 Points**]

Solution:

b) Assuming that memory is byte addressable and addresses are 35-bits long, give the number of bits required for each of the following fields: [**6 Points**]

TAG:
SET INDEX:
BLOCK OFFSET:

c) What is the total size of the cache in bits, including the valid, tag and data fields? [**3 Points**]

Solution:

d) Assume that the Trashium cache communicates with main memory via a 64-bit bus that can perform one transfer every 10 cycles. Main memory itself is 64-bits wide and has a 10-cycle access time. Memory accesses and bus transfers may be overlapped. What is the miss penalty for the cache? In other words, how long does it take to send a request to main memory and to receive an entire cache block? Explain. [**4 Points**]

Solution:

e) Assume the memory system described above, if the cache has a 95% hit rate and a one-cycle hit time, what is the average memory access time? [**3 Points**]

Solution:

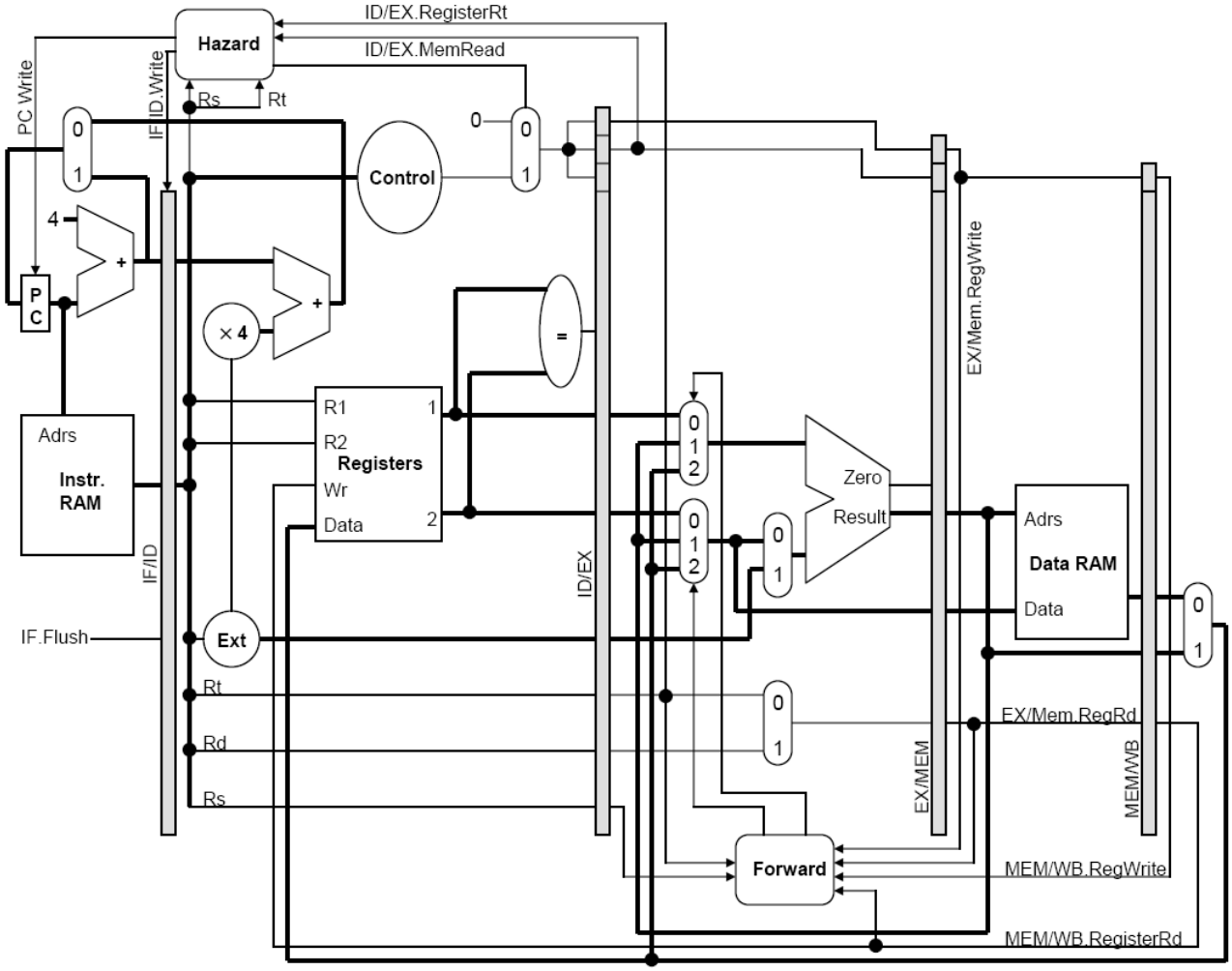
f) If we run a program which consists of 30% load/store instructions, what is the average number of memory stall cycles per instruction assuming again the memory system above? [**3 Points**]

Solution:

Problem 6: **(8 parts) Pipelining Performances Including Caches** [**42 Points**]

Assume that the following MIPS assembly code is run of the pipelined datapath shown below.

```
Frodo:
     lw    $t0, 0($a1)
     mul   $t1, $t0, $a2
     lw    $t0, 0($a0)
     add   $t0, $t0, $t1
     sw    $t0, 0($a0)
     sub   $a3, $a3, 1
     addi  $a0, $a0, 4
     addi  $a1, $a1, 4
     bne   $a3, $0, Frodo
```



a) Find the number of clock cycles needed to execute this code, accounting for all possible stalls and flushes. Assume that **$a3** is initially set to 100. Explain. [**5 Points**]

Solution:

(Problem 6 cont'd)

b) Rewrite the code to eliminate as many stalls as possible. Show your modified assembly language program below. [**5 Points**]

Solution:

c) What is the performance improvement of your new function as compared to the original one? Again, assume that **`$a3`** is initially set to 100. [**5 Points**]

Solution:

d) Suggest a way to rewrite this code to reduce the number of cycles lost to flushes. You do not have to show any actual code. [**5 Points**]

Solution:

(**Problem 6 cont'd**)

Assume you run the Frodo code on a Corleone2000 processor that has two levels of data caches, with the characteristics shown below. Assume that it takes 50 clock cycles to request and complete a 32-byte transfer between main memory and the L2 cache.

	L1	L2
Data size	32 KB	256 KB
Block size	8 bytes	32 bytes
Associativity	Direct-mapped	4-way
Hit time	1 cycle	19 cycles
Miss rate	5%	2%

e) How many bits of a 32-bit main memory address would be used as the tag, index and block offset fields, for each of the two caches above? [**6 Points**]

	L1	L2
Tag size		
Index size		
Block offset		

f) Find the average memory access time for both the L2 and L1 caches. Explain. [**5 Points**]

Solution:

g) Looking back at the Frodo code, assume that **`$a0`** and **`$a1`** are initially 3000 and 4000, and the loop iterates 100 times. Both level 1 and level 2 data caches are empty at first. How many memory stall cycles would be required for the execution of this code? [**6 Points**]

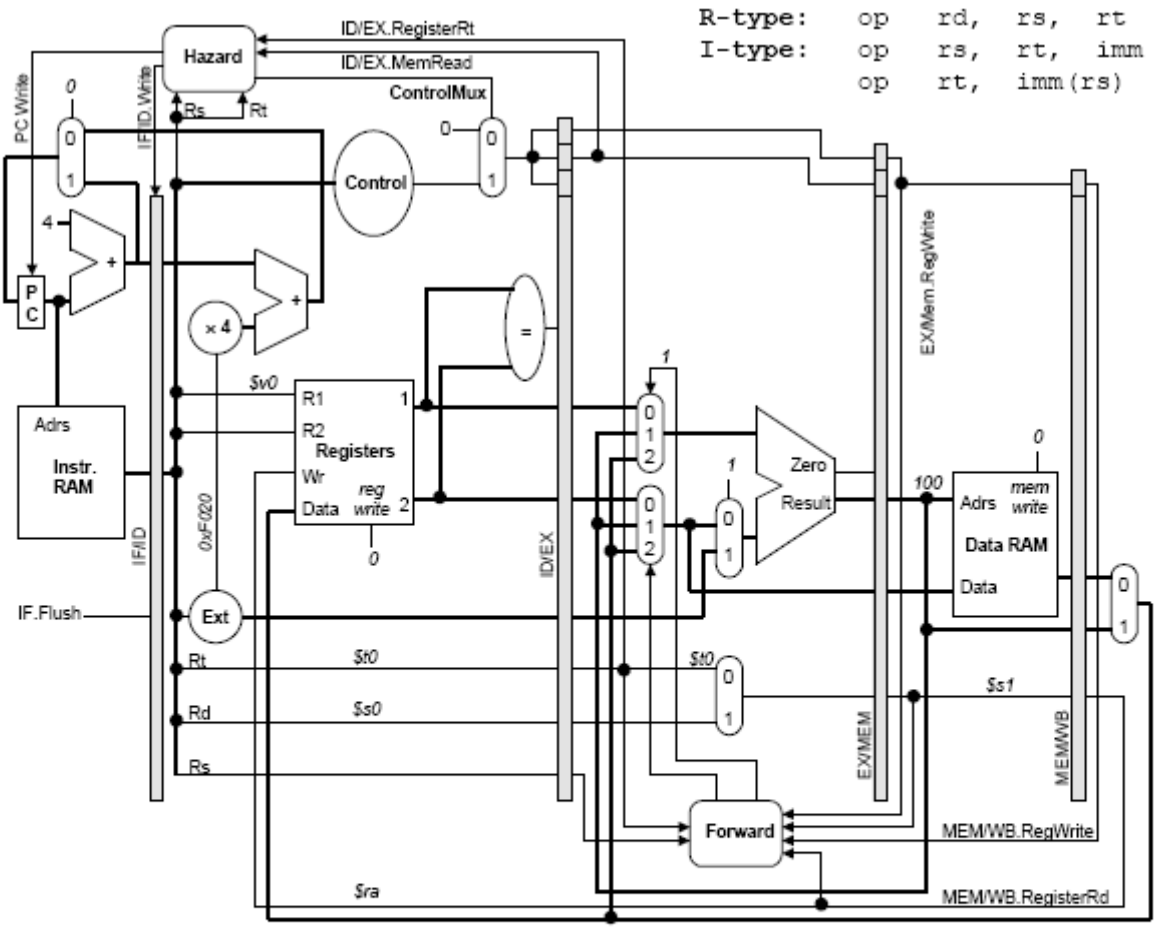
Solution:

h) Briefly discuss how each of the different write policies (write-back or write-through for hits, and allocate-on-write or write-around for misses) might affect the performance of the Frodo function. [**5 Points**]

Solution:

Problem 7: **More on Pipelining** [**16 Points**]

Below is a MIPS pipelined datapath, annotated with some signal values. Select the instructions in each pipe stage that are consistent with the annotated signal values. Be careful, these are tricky because the datapath includes some values that are correct, but will be ignored.



IF stage

ID stage

a) add $t0, $s0, $v0
b) bne $t0, $s0, label
c) bne $s0, $v0, label
d) add $s0, $v0, $t0
e) bne $v0, $t0, label

EX stage

a) sub $t0, $s8, $t0
b) addi $t0, $s0, 100
c) subi $s1, $t0, 100
d) addi $a2, $s1, 100
e) sub $t0, $t0, $s1

MEM stage

WB stage

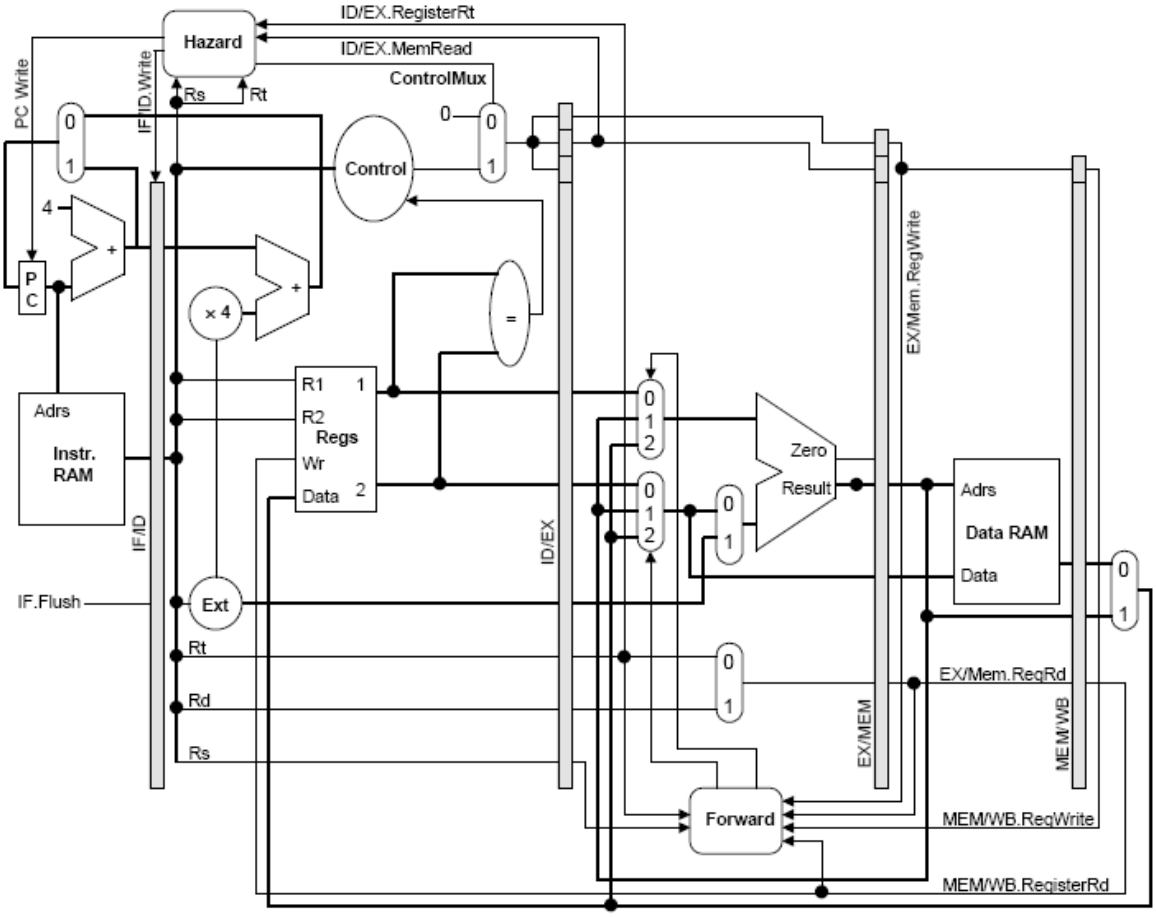
a) sw $t0, -4($v0)
b) addi, $ra, $v0, 100
c) lw $a1, 16($s1)
d) sll $ra, $s1, $t0
e) sub $s1, $ra, $a3

Problem 8: **Pipelining and Data Hazards** [**18 Points**]

Consider again the pipelined MIPS datapath shown below where branches are resolved in the ID stage. Assume we want to change the format of the branch instruction to

```
beq $t1,$t2,$t3,    # PC← $t3 if $t1=$t2
```

where **`$t3`** now contains the branch target address. Modify the datapath so that it supports the new branch format [**4 Points**], and then add forwarding data paths to correctly support all data hazards for the new **`beq`** instruction [**8 Points**]. Write the control logic for the forwarding muxes you use. [**6 Points**]



Control equations: