University of Illinois at Urbana-Champaign
Department of Computer Science

Final Examination

CS232 Fall 2000

Name: Solutions
(last) (first)

NetId: ________________

Question	Max Score	Grader	Score
1	23		
2	5		
3	15		
4	15		
5	10		
6	10		
7	20		
8	10		
9	10		
10	10		
11	12		
Total	140		

NO CALCULATORS, CLOSED BOOK

1. Multiple Choice Problems

A) [5] Choose the one untrue statement about MIPS addressing modes:

   a. Pseudo-direct addressing, used for jumps, gets the upper 4 bits of the address from the PC
   b. Base addressing adds an offset to a base address
   c. Immediate addressing gets its operand from the instruction
   d. PC-relative addressing, used for branches, gets the upper 16 bits of the address from the PC
   e. Register addressing gets its operands from registers

B) [9] For each of the MIPS code sequences below, indicate if the memory accesses show temporal locality, spatial locality, both, or neither (by checking the corresponding boxes).

a.

```
       li   $t0, 100
LOOP:  lw   $t1, 32767($s0)
       lw   $t2, 0($s0)
       addi $t0, $t0, -1
       bne  $t0, zero, LOOP
```

[X] temporal locality, [ ] spatial locality, [ ] both, [ ] neither

B)
3 points for each on first sub-problem (a.), give 1 point for "both".

b.

```
        li   $t0, 100
LOOP1:  li   $t1, 4
LOOP2:  add  $t2, $s0, $t1
        lb   $s1, 0($t2)
        addi $t1, $t1, -1
        bne  $t1, zero, LOOP2
        addi $t0, $t0, -1
        bne  $t0, zero, LOOP1
```

[ ] temporal locality, [ ] spatial locality, [X] both, [ ] neither

c.

```
       li   $t0, 100
LOOP:  lw   $s1, 0($t0)
       addi $t0, $t0, -1
       bne  $t0, zero, LOOP
```

[ ] temporal locality, [X] spatial locality, [ ] both, [ ] neither

C) [9] The following problems relate to the various characteristics of a cache.

a) For each of the following characteristics of a cache, mark if it's a good characteristic, or a bad one.

1 point each

i. decreasing miss rate is (**good** / bad).
ii. increasing the miss penalty is (good / **bad**).
iii. increasing the amount of tag storage with respect to the amount of data storage is (good / **bad**).

b) Mark how the characteristics would change if we increased the block size.
i. the miss rate, in general, (increases / **decreases**).
ii. the miss penalty (**increases** / decreases).
iii. when the block size gets more and more closer to the cache size, the miss rate starts to (**increase** / decrease).
iv. the amount of tag storage with respect to the amount of data storage (increases / **decreases**).

c) Mark how increasing the associativity of the cache would affect its behavior.
i. the miss rate (increases / **decreases**).
ii. the hit time generally (**increases** / decreases).

Short Answer Problems

2. [5] Write 2.5 in the standard IEEE 754 single precision floating point binary representation.

0	10000000	01000000000000000000000

Point breakdown:
1 – sign
2 – exponent (1 partial, if know the concept of bias)
2 – significand (1 partial, if know where the significand should go, at least)
-1 – binary conversion mistake

3. [15] Write down a sentence on the characteristic feature of each of the following datapath implementations that differentiate one from the other.

The number to the left is the points given for the particular answer. If the student writes several of the items, the points are summed up until the total reaches 3, the maximum points for a sub-problem.

Single cycle:

2 – One instruction takes one cycle.
3 – Clock cycle time is the maximum of the execution times of the instructions.
2 – There are separate functional units for each stage within an instruction.

Multi-cycle:

3 – number of cycles vary for each instruction, and therefore performace is better.
3 – describes FSM in detail (2 if not in detail)
2 – functional units are combined
1 – says something about "many stuff"
1 – the cycles are shorter

Pipelined:

3 – different instructions in different execution stages run simultaneously
1 – just "more than one instruction at a time"

Superscalar:

3 – execute multiple instructions in the same cycle.
1 – load/stores and ALU instructions done simultaneously

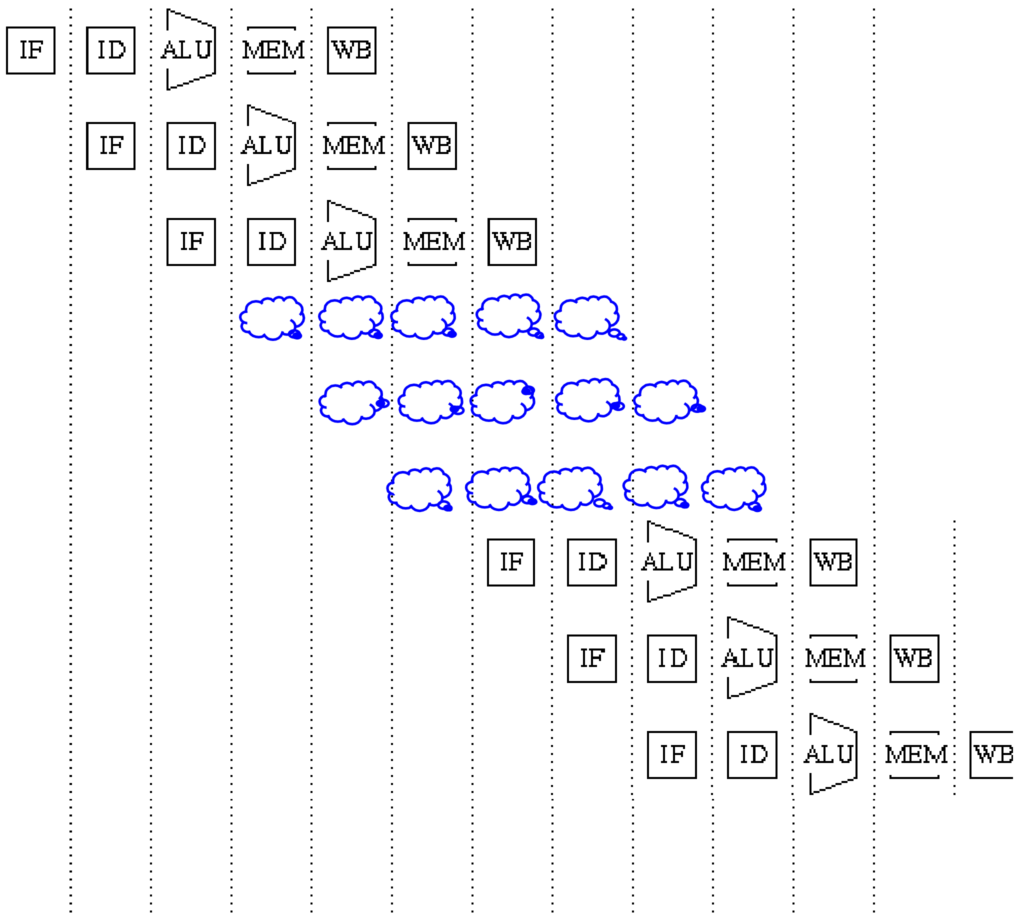
Superpipelining:

3 – just more stages to pipelining

Long Answer Problems

4. [15] Consider a modification to the register file in the pipelined MIPS datapath. This modification allows the register file to do only one operation in each cycle: i.e. it can be either read from or written to in a single cycle.

   This modification introduces structural hazards into MIPS. Show how a sequence of instructions that have NO data or control hazards would now be executed using the following diagram. Show atleast the next five instructions



5. [10] Consider a simple, **non-pipelined**, **single-cycle** implementation of MIPS. Assume instruction fetch from memory takes the same time as data fetch from memory. The operation time for the major functional units for this machine are as follows:
- Memory units: 6ns
- ALU and adders: 4ns
- Register file (read or write): 2ns

(a) [5] Put a mark in the each blank to indicate the stages of the critical path taken by the following instructions.

1 point each

Instruction Type	**Functional Units used by the Instruction Type**				
	Instruction fetch	Register access	ALU	Data Access	Write Register
R-format (ALU)	X	X	X		X
Load word	X	X	X	X	X
Store word	X	X	X	X	
Branch	X	X	X		
Jump	X				

(b) [2] Assuming that no other delays arise from the memory system, internal control unit, program counter access, etc., find the clock cycle time for the single cycle datapath. Write a line about how you determined the clock cycle time.

The longest instruction is load word, and it takes 6+2+4+6+2=20ns. So, 20ns is the clock cycle time.

1 for 20ns
1 for explanation

(c) [3] For the given an instruction mix compare the performance of the **pipelined** and **multi-cycle** implementation of the datapath above

- Stores: 12% $\times$ **18 = 2.16**
- ALU: 44% $\times$ **14 = 6.16**
- Branch: 18% $\times$ **12 = 2.16**
- Jump: 2% $\times$ **6 = 0.12**
- Load 24% $\times$ **20 = 4.8**

+__________

1 for multicycle performance
1 for pipelined performance
1 for conclusion
(credit also given for comparison using CPI, since cycle time is the same)

15.40 (Multicycle Avg time per instruction)
Since, slowest functional unit takes 6ns, if ignoring stalls, hazards, etc., avg time per instruction for pipelined implementation is 6ns.
So, the pipelined implementation is faster by 15.40/6.

6. [10] Given the C code below, fill in the blank lines in the MIPS code below so it will execute the C code properly. You may use only the blanks provided.

C code:

```
int x,y,err;
//the code to input the value of x is not shown

if(x<10)
   err = 0;
else {
   x = 0;
   err = 1;
}
y = x;
```

MIPS code:

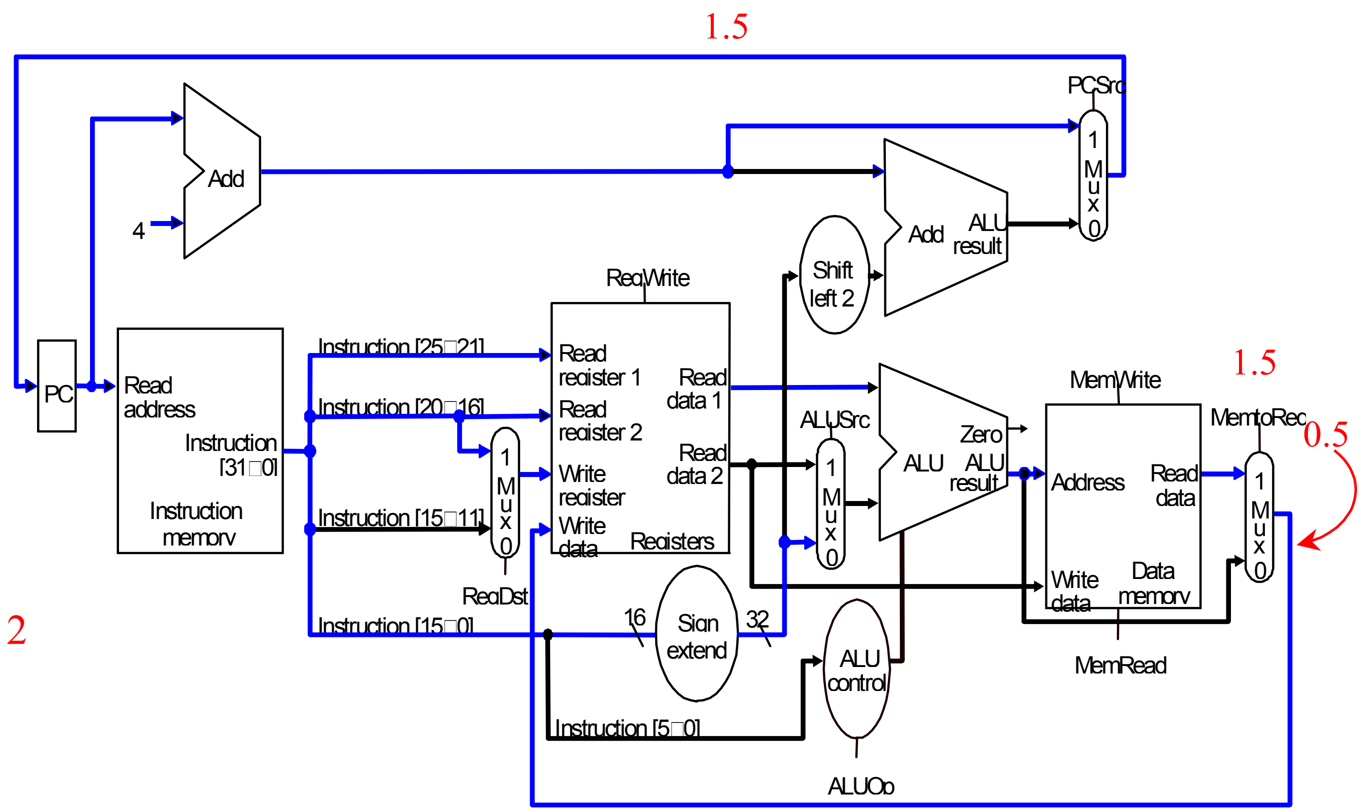
Labels are provided at the beginning of each line so that you do not have to add any labels.

```
#$s0 = int x;
#$s1 = int y;
#$s2 = int err;

#not shown here is the code that inputs a value for x

L1:  slti $t0, $s0, 10
     ________________
L2:  beq $t0, $0, L5

L3:  add $s2, $0, $0

L4:  beq $0, $0, L7
     ________________
L5:  add $s0, $0, $0

L6:  addi $s2, $0, 1

L7:  add $s1, $s0, $0
     ________________
```

Point Breakdown:
- Fully correct → 10
- Logic → 3
- Syntax → 2
- 2 points per each blank (Syntax → 1)

7. [20] Consider the single-cycle datapaths shown below, highlight **clearly** the paths to show the flow of both the **data** and the **PC** for the instructions given, then fill in the blanks for the control bits (0 = cleared, 1 = set, x = don't care). For the ALUOp field, use the following: 00 = "add", 01 = "subtract", 10 = "do the funct field operation".

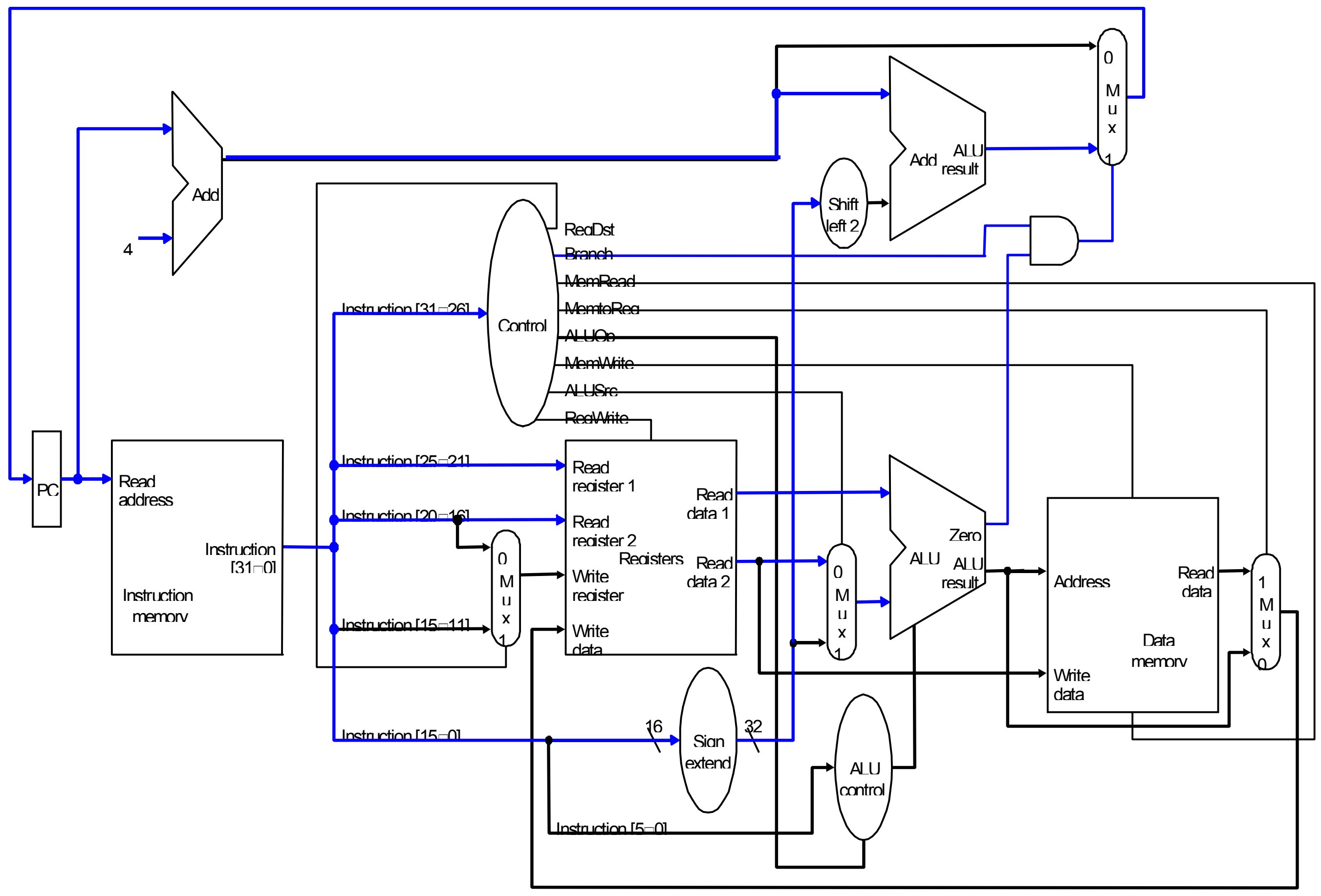
(a) [10] **Instruction**: `lw $1, 100($2)`



PCSrc	RegDst	RegWrite	ALUSrc	ALUOp	MemWrite	MemRead	MemtoReg
1	1	1	0	00	0	1	1

8

(b) [10] **Instruction:** `beq $2, $2, loop`



Branch	RegDst	RegWrite	ALUSrc	ALUOp	MemWrite	MemRead	MemtoReg
1	X	0	0	01	0	X	X

8. [10] Referring to both of the diagrams from problem number 7, assume that the operation time for the major functional units are following:
   - Memory units: 3ns
   - Register file (read or write): 1 ns
   - General ALU: 2ns
   - Adder for PC + 4: X ns
   - Adder for branch address computation: Y ns
   - Assume other units have no delay

   (a) [5] If X = 4 ns, Y = 4 ns, what are the cycle times for the lw and beq instructions?

   cycle time for beq:

   2 points 4 + 4 = 8 ns

   cycle time for lw:

   2 points 3 + 1 + 2 + 3 + 1 = 10 ns

   What would be the cycle time of a datapath that supports only these two instructions?

   1 points 10 ns

   (b) [5] If X = 2ns, and Y = 10ns, what are the cycle times for the beq and lw instructions?

   cycle time for beq:

   13 ns (takes 10ns + waiting for 2 inputs takes max(3,2)=3ns)

   cycle time for lw:

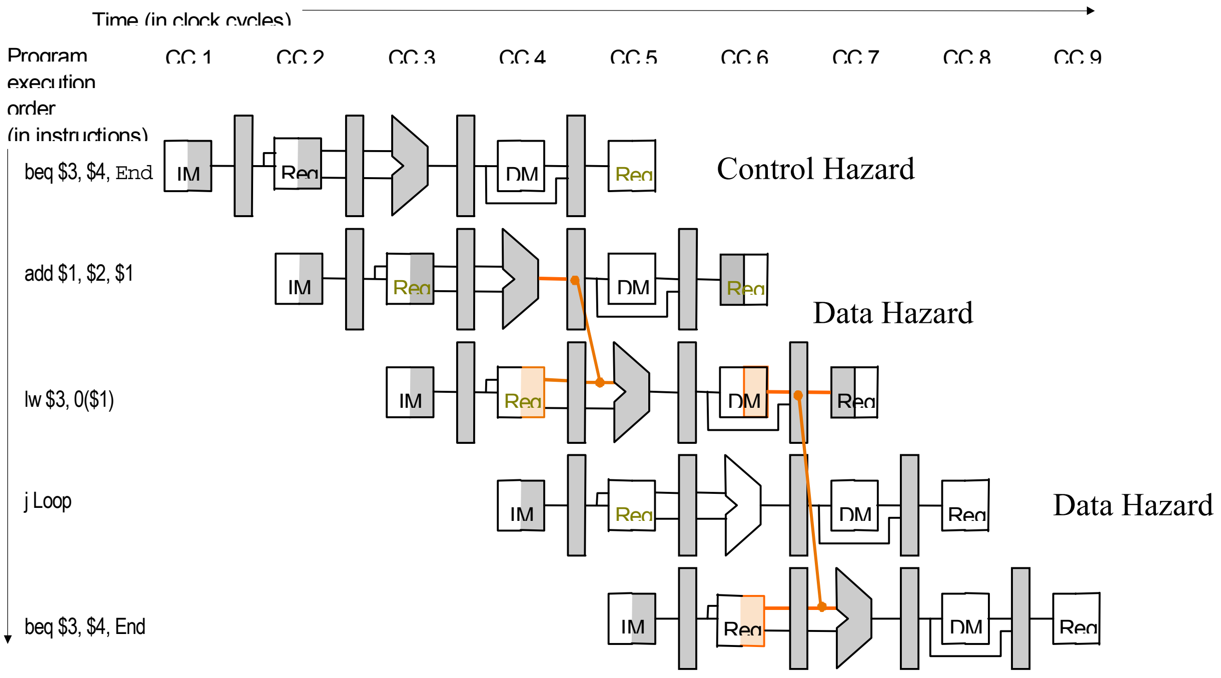
   10 ns

   What would be the cycle time of a datapath that supports only these two instructions?

   10 ns

9. [10] Identify all the hazards in the following code, and **state the type of each hazard**.

```
Loop:
         Beq $3, $4, End      control
         Add $1, $2, $1
data                    data
         Lw  $3, 0($1)
         J Loop
End:
```



10. [10] Given two pieces of code, the recursive fibbonachi code (MP1 – used recursion and function return values) and the memoizing fibbonachi code (MP2 – filled in an array recursivly) we wrote, **compare the performance of the two machines described below**. Ignore the instructions that are used to maintain the stack. Both machines run at the same clock rate

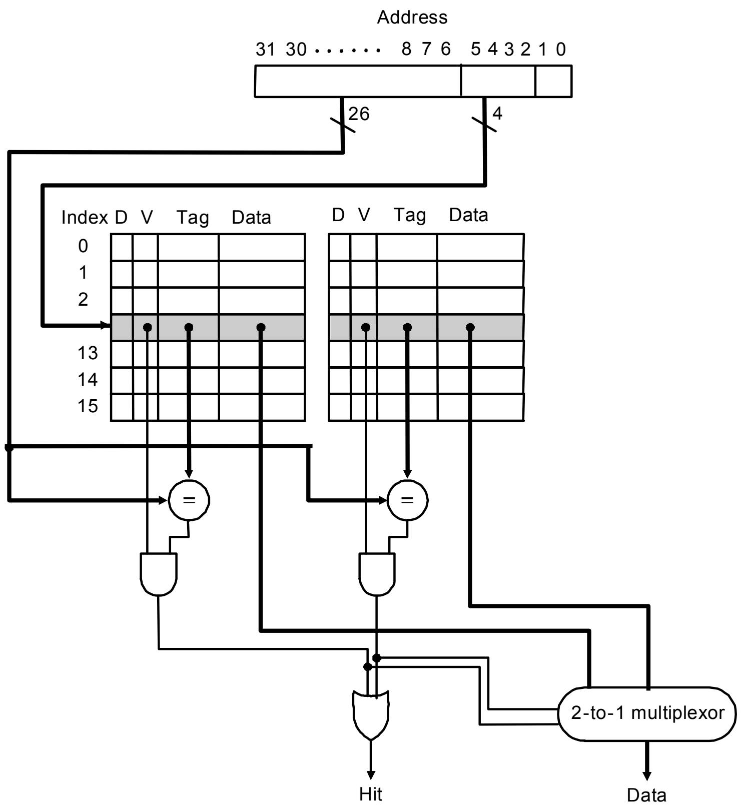
Machine A – superscalar MIPS machine that executes both one load/store and one ALU/branch instruction per cycle.

Machine B – pipelined MIPS machine that completes one instruction per cycle.

Recursive fibbonachi: Same speed

Array-based fibbonachi: Superscalar is faster

11. [12] Consider a set-associative cache as given in the diagram below.



We are using a write-back policy on this cache, and we are using an extra dirty bit ("D") to mark those blocks that actually need to be written back when replaced, similar to what is done in virtual memory. Note that the dirty bit is examined on each replacement. The replacement policy is LRU(i.e. replace the least recently used block). Assuming that the cache is empty at the beginning, show how the cache changes for the following code sequence. There is a set of tables on the next page for this purpose. (continued on next page)

Note that only the first two blocks of the cache are shown. Ignore what happens in the other blocks. You must fill in the "D", "V", and tag fields. Leave the field for the data blank. You may omit the leading 0's for the values in the tag fields. Also, if there is a memory access, write whether the access is a read or write, and then write the address.

a. `lw $t0, 01000100`$_{two}$`($0)`

Index	D	V	Tag	Data	D	V	Tag	Data
0								
1	0	1	1					

Memory Access? : Read 01000100

b. `sw $t0, 10000100`$_{two}$`($0)`

Index	D	V	Tag	Data	D	V	Tag	Data
0								
1	0	1	1		1	1	10	

Memory Access? : none

c. `lw $t1, 01000100`$_{two}$`($0)`

Index	D	V	Tag	Data	D	V	Tag	Data
0								
1	0	1	1		1	1	10	

Memory Access? : none

d. `lw $t2, 11000100`$_{two}$`($0)`

Index	D	V	Tag	Data	D	V	Tag	Data
0								
1	0	1	1		0	1	11	

Memory Access? : write 10000100, read 11000100

e. `sw $t1, 11000000`$_{two}$`($0)`

Index	D	V	Tag	Data	D	V	Tag	Data
0	1	1	11					
1	0	1	1		0	1	11	

Memory Access? : none

Memory accesses: 1 for each line
D, V flags : 1 if all correct
Tags : 2 if all correct
Placement of data : 4 (partial credit depending on how much understanding shown)