CSEE 3827: Fundamentals of Computer Systems

Final Exam

May 10, 2010

Name: SOLUTION UNI: ______________

Read all of the following information before starting the exam:

- Be sure to write your name on each page of the exam.
- Use the exam itself for your solutions (no blue books or spare sheets of paper). You may use the backside of pages if you need more space.
- Show your work in order to earn partial credit.
- You may use your textbook and class notes, but *absolutely no electronic devices (laptops, cell phones, etc.)*
- Good luck!

Problem	Point Value	Points Earned
1	4	
2	4	
3	4	
4	4	
5	4	

Problem	Point Value	Points Earned
6	4	
7	8	
8	8	
9	12	
10	16	
Total	68	

1. **(4 points)** True or false: The control units (main and ALU) for a single cycle data path are sequential logic components.

False.

2. **(4 points)** Place either + or - in each quadrant below to indicate whether the listed feature (hardware or software design) is a plus or a minus of CISC and RISC architectures.

	RISC ISAs	CISC ISAs
hardware design	+ (simple hardware)	— (complex hardware)
software design	— (increased instr. count)	+ (more powerful instructions)

3. **(4 points)** For each instruction in the following code sequence, indicate by circling the stage name, in which stage of a 5-stage MIPS pipeline it will be during the fifth clock cycle. Assume that the pipeline has full forwarding (MEM-EX and WB-EX) and hazard detection. Also assume in the first cycle that the processor fetches the first instruction in the sequence.

`lw $t0, 0($s0)`	IF ID EX MEM (WB)
`add $s0, $t1, $t2`	IF ID EX (MEM) WB
`lw $t1, 4($s0)`	IF ID (EX) MEM WB
`sub $s0, $t1, $t1`	IF (ID) EX MEM WB

4. **(4 points)** For the MIPS designs we've studied, please indicate whether each of the following modifications would *improve*, *degrade*, have *no effect* or have an *indeterminate* effect on the latency and throughput of the processor.

	Latency	**Throughput**
Reducing the cycle time of a single-cycle datapath	*improve*	*improve*
Increasing CPI, keeping clock rate constant	deg.	deg.
Changing a single-cycle design running at X MHz to a 5-stage pipeline running at $\frac{X}{5}$ MHz	deg.	deg.
Removing the forwarding paths from a pipelined datapath	deg.	deg.

5. **(4 points)** Which of the following four memory reference streams has the most spatial locality? Which has the greatest temporal locality? Briefly (i.e., less than one sentence) justify your choices.

(a)	(b)	(c)	(d)
0x0000	0x0000	0x0000	0x0000
0x0004	0x0001	0x0001	0x0000
0x0008	0x0002	0x0000	0x0000
0x000c	0x0003	0x0001	0x0000
0x0010	0x0004	0x0000	0x0000
0x0014	0x0005	0x0001	0x0008
0x0018	0x0006	0x0000	0x0008
0x001c	0x0007	0x0001	0x0008
0x0020	0x0008	0x0000	0x0008

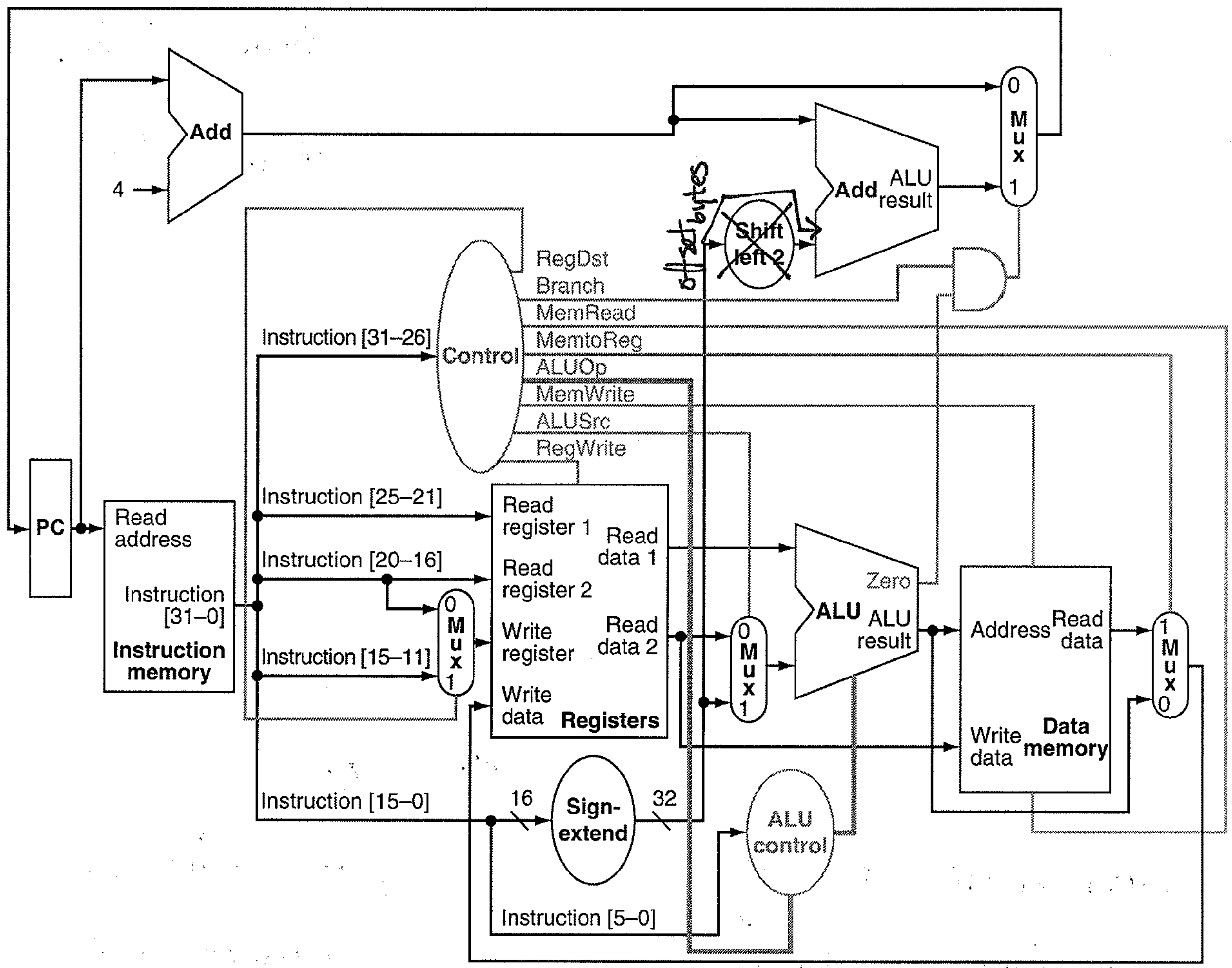
(c): most spatial locality — only two addresses accessed and they're adjacent

(d): most temporal locality — longest sequences of repeated accesses to same address

6. **(4 points)** In MIPS, the branch target is given in terms of instruction offset from the PC following the branch instruction ($PC_{target} = PC_{branch} + 4 + (offset_{ins} \times 4)$).

   (a) If the branch target were instead based on $offset_{bytes}$, the number of bytes between the PC of the branch and target, what would the new formula for PC_{target} be? (Note: this change would allow programs to jump to non-word-aligned addresses, e.g., to jump 3 bytes ahead, although for most programs $offset_{bytes}$ would be a multiple of 4.)

   (b) How would the branch target calculation hardware in the datapath below need to change to accomodate this new system?



a) $offset_{ins} \times 4 = offset_{bytes}$

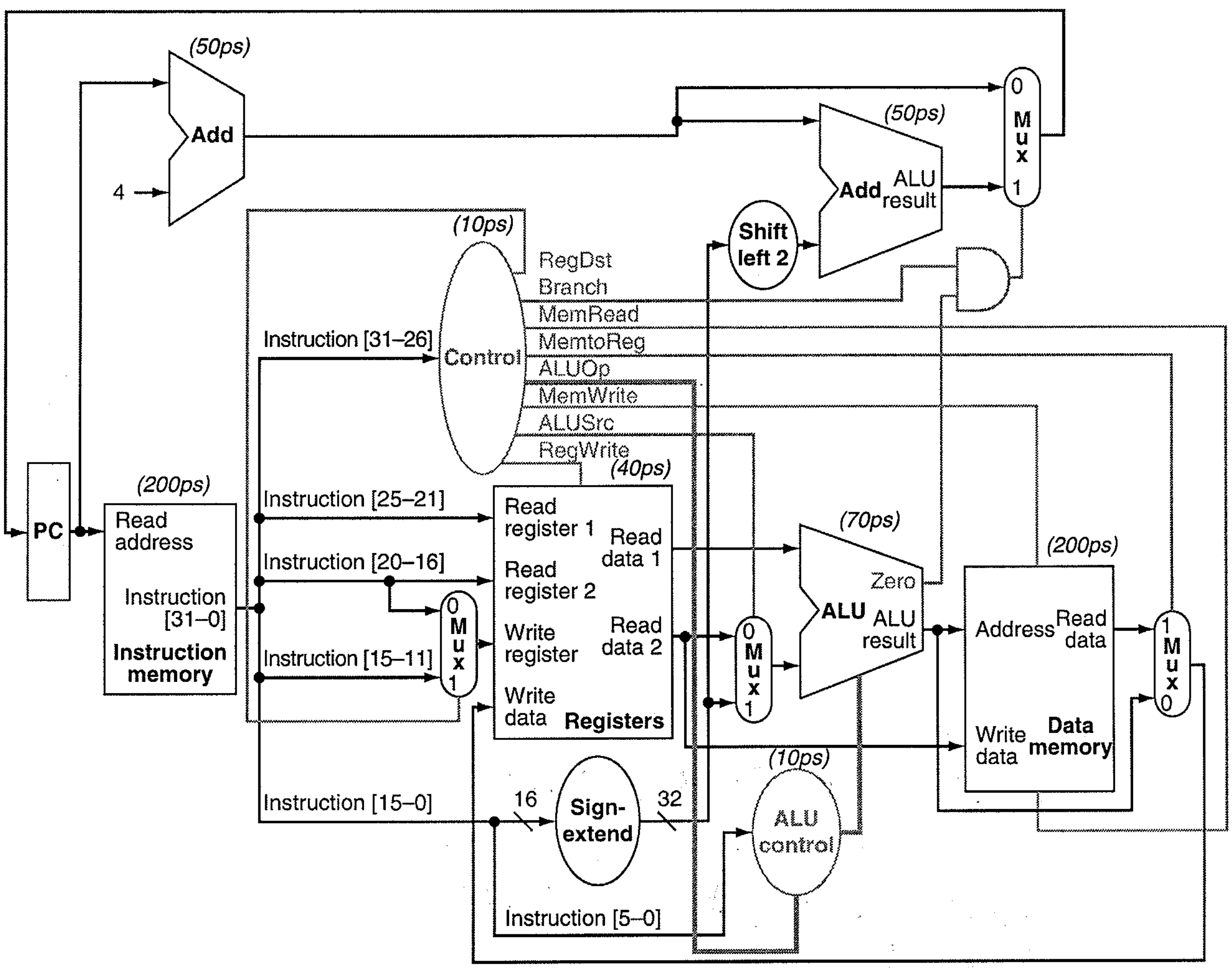
$PC_{target} = PC_{branch} + 4 + offset_{bytes}$

b) Since the immediate field contains $offset_{bytes}$ just feed that into the next PC adder.

7. **(8 points)** The single-cycle MIPS processor below has been annotated with delays for several of the modules (e.g., the register file has a 40ps delay). Assuming the miscellaneous, unlabeled modules have zero delay, give the critical path delay for the following instructions:

- a *load*
- a *store*
- an *R-Type* instruction
- a *beq*

Given these critical paths, what is the maximum frequency for this datapath?



load 200+40+70+200+40 = 550 ps

store 200 + 40 + 70 + 200 = 510 ps

R-type 200 + 40 + 70 + 40 = 350ps

beq 200 + 40 + 70 = 310 ps

Max delay = 550ps

Max frequency = $\frac{1}{550 \text{ ps}} \approx 1.8$ GHz

8. **(8 points)** In this problem you will implement a hex to decimal converter in MIPS assembly. Your function should be called `hex2dec`. It takes as an argument a single ASCII character, and returns its decimal value.

For example, the decimal value of the hex digit 'A' is 10. If the function is called on 'A', the input argument would be $arg = 0x00000041$ (the ASCII code for 'A') and the output should be $ret = 0x0000000a$ (the value 10).

- You may assume that it is called only on valid inputs: 0-9,A-F.
- Be sure your implementation adheres to proper MIPS calling conventions.
- You need write a full program, just write the code for the `hex2dec` function.

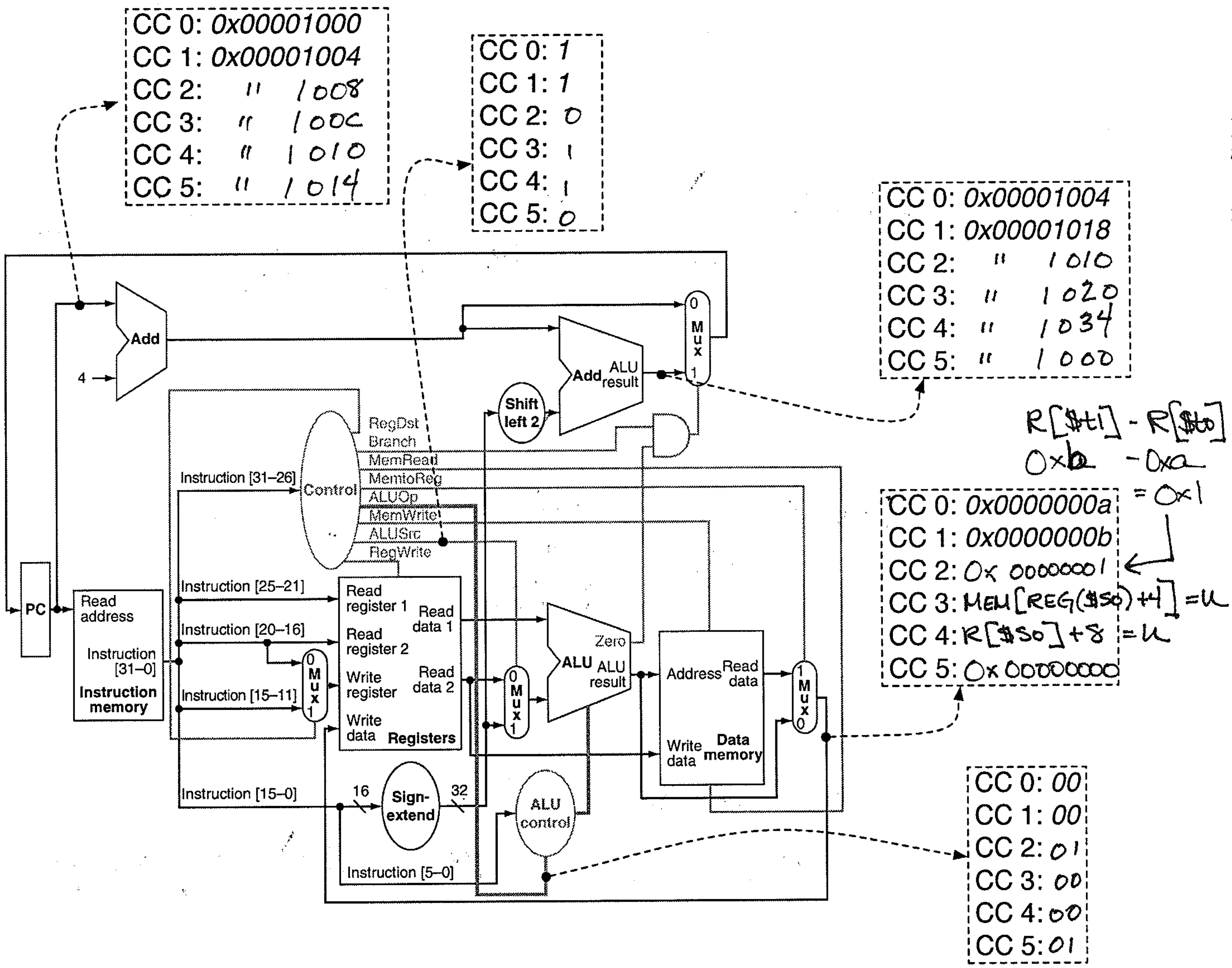
For reference, here are the ASCII codes for the relevant characters:

Input: Hex Digit		Output: Decimal Value	
Symbol	ASCII code (hex)	(decimal)	(hex)
0	**0x30**	0	**0x0**
1	**0x31**	1	**0x1**
2	**0x32**	2	**0x2**
3	**0x33**	3	**0x3**
4	**0x34**	4	**0x4**
5	**0x35**	5	**0x5**
6	**0x36**	6	**0x6**
7	**0x37**	7	**0x7**
8	**0x38**	8	**0x8**
9	**0x39**	9	**0x9**
A	**0x41**	10	**0xa**
B	**0x42**	11	**0xb**
C	**0x43**	12	**0xc**
D	**0x44**	13	**0xd**
E	**0x45**	14	**0xe**
F	**0x46**	15	**0xf**

```
hex2dec:
    slti  $t0, $a0, 0x41
    beq   $t0, $zero, letter
    addi  $v0, $a0, -0x30
    j end
letter:
    addi  $t0, $a0, -0x41
    addi  $v0, $t0, 10
end:
    jr $ra
```

9. **(12 points)** For the processor below, fill in the blanks giving the values on the indicated wires for cycles 2-5. Cycles 0 and 1 are provided to get you started. You may use "U", if necessary, for undefined values.

```
TOP: lw $t0, 0($s0)          # opcode=0x23, rs=0x10, rt=0x08, imm=0x0000
     lw $t1, 4($s0)          # opcode=0x23, rs=0x10, rt=0x09, imm=0x0004
     beq $t0, $t1, LBL       # opcode=0x04, rs=0x08, rt=0x09, imm=0x0001
     sw $t0, 4($s0)          # opcode=0x2b, rs=0x10, rt=0x08, imm=0x0004
LBL: addi $s0, $s0, 8        # opcode=0x08, rs=0x10, rt=0x10, imm=0x0008
     beq $zero, $zero TOP    # opcode=0x04, rs=0x00, rt=0x00, imm=0xfffb (-6 decimal)
```



10. **(16 points)** This problem examines the following MIPS implementation of a priority encoder. The instructions have been labeled (i1 - i7) for convenient reference.

```
priority_encode:
(i1)    addi $t0, $zero, 0          # counter of position

priority_loop:                      # shifts value until eq zero (shifts - 1 is answer)
(i2)    srl $a0, $a0, 1
(i3)    beq $a0, $zero, priority_end
(i4)    addi $t0, $t0, 1            # increment for next shift
(i5)    j priority_loop

priority_end:
(i6)    add $v0, $zero, $t0         # load value into return value register
(i7)    jr $ra
```

(a) Assume any call to `priority_encode` returns the value N. Give an expression for the number of instructions executed as a function of N.

instrs = 1 (i1) + 4N (i2-i5) + 2 (i2, i3) + 2 (i6, i7) = 4N+5

(b) Give an expression for the number of `addi` instructions as a function of N.

addis = 1 (i1) + 1·N (i4 per loop) = N+1

(c) Assuming $N = 2$, how many cycles would it take for the function to complete on a single-cycle processor?

1 instr per cycle

4N+5 = 4·2+5 = 13 cycles

(d) Still assuming $N = 2$, On the standard, 5-stage MIPS pipeline, with full forwarding (i.e., bypass paths from MEM to EX and from WB to EX), indicate, when, if ever, the processor would need to insert bubbles. Answers should be of the form "between iX and iY".

Between i3 and i6 or i3 and i4 depending on which way the beq goes. Two bubbles.

(e) On the pipelined processor, how many cycles will it take for a call where $N = 2$ to complete?

cycles = 13 instrs + 3 bubbles + 4 cycles to fill pipe

↑ beq exe three times

= 20 cycles

(f) Assume the single cycle processor runs at 100MHz, while the pipelined version runs at 500MHz. Which processor would execute the call where $N = 2$ faster?

single cycle: $T_{sc} = 13 \frac{\text{instr}}{\text{prog}} \cdot 1 \frac{\text{cycle}}{\text{instr}} \cdot \frac{1}{100 \times 10^6} \frac{\text{sec}}{\text{cyc}}$

$= \frac{13}{100 \times 10^6} \frac{\text{sec}}{\text{prog}}$

pipelined: $T_{pipe} = 20 \frac{\text{cyc}}{\text{prog}} \cdot \frac{1}{500 \times 10^6} \frac{\text{sec}}{\text{cyc}}$

$= \frac{20}{500 \times 10^6} \frac{\text{sec}}{\text{prog}}$

← smaller value, so pipelined processor is faster