CS232 Exam 2
March 16, 2007

Name: _____

Section: 10am noon 2pm 4pm (circle one)

- This exam has 5 pages, including this cover.
- There are three questions, worth a total of 99 points; plus 1 point for the cover question.
- No written references or calculators are allowed.
- Write clearly and show your work. State any assumptions you make.
- You have 50 minutes. Budget your time, and good luck!

Question	Maximum	Your Score
1	30	
2	45	
3	24	
Cover	1	
Total	100	

Cover Question: How big is a byte? _____8_____ bits (1 point)

Question 1: Single-cycle CPU implementation (30 points)

For this question, we will implement a hypothetical instruction **sw+** in the single-cycle pipeline. **sw+** is a "store word, with post increment" that is found in some real architectures (e.g., IA-64). It is encoded as an I-type instruction and performs the following operations:

M[R[rs]] = R[rt]
R[rs] = R[rs] + imm

Field	op	rs	rt	imm
Bits	31-26	25-21	20-16	15-0

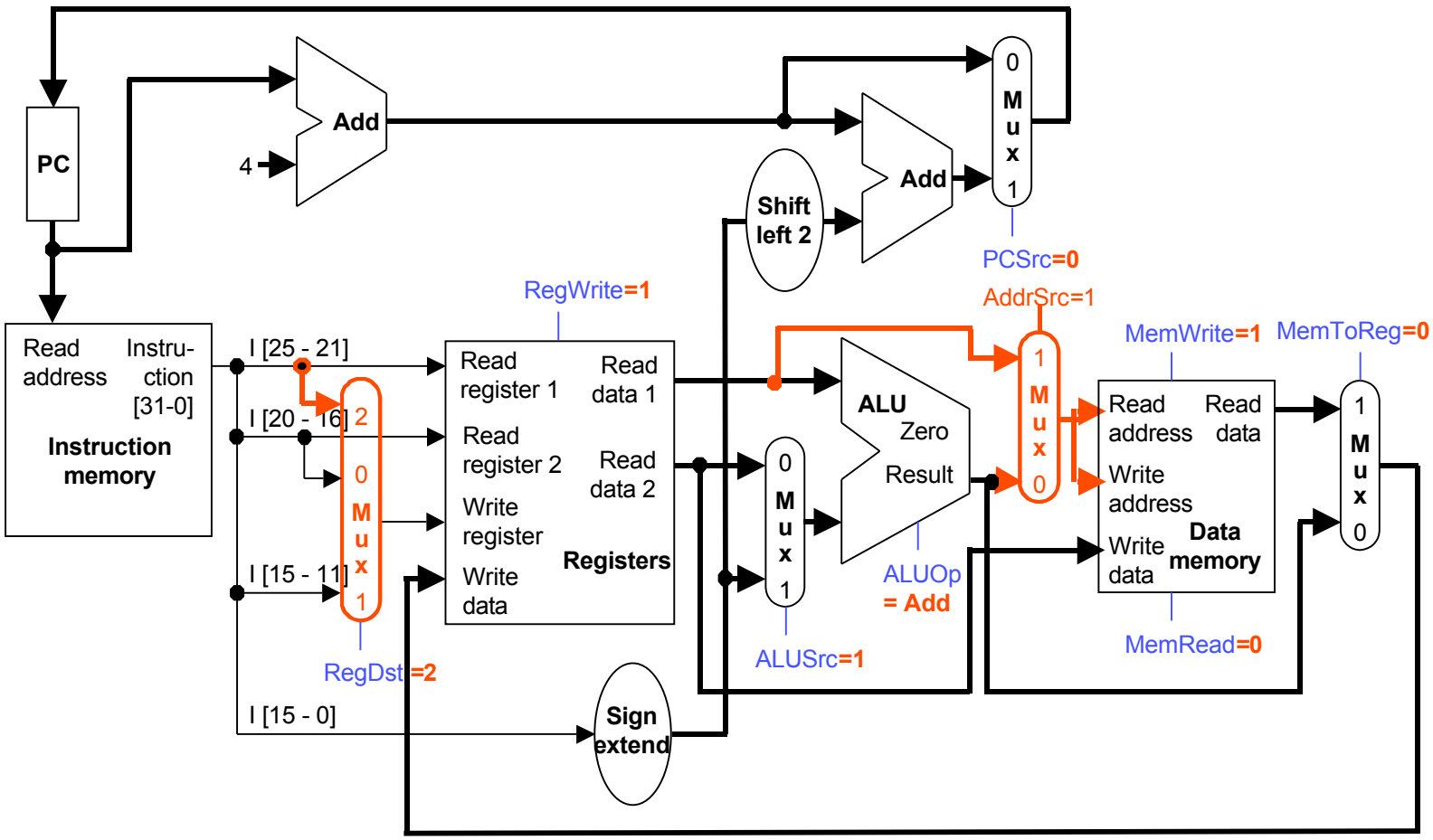
Part (a)

The single-cycle datapath from lecture appears below. Show what changes are needed to support **sw+** instruction. You should only add wires and muxes to the datapath; do not modify the main functional units themselves (the memory, register file and ALU). Try to keep your diagram neat! (20 points)

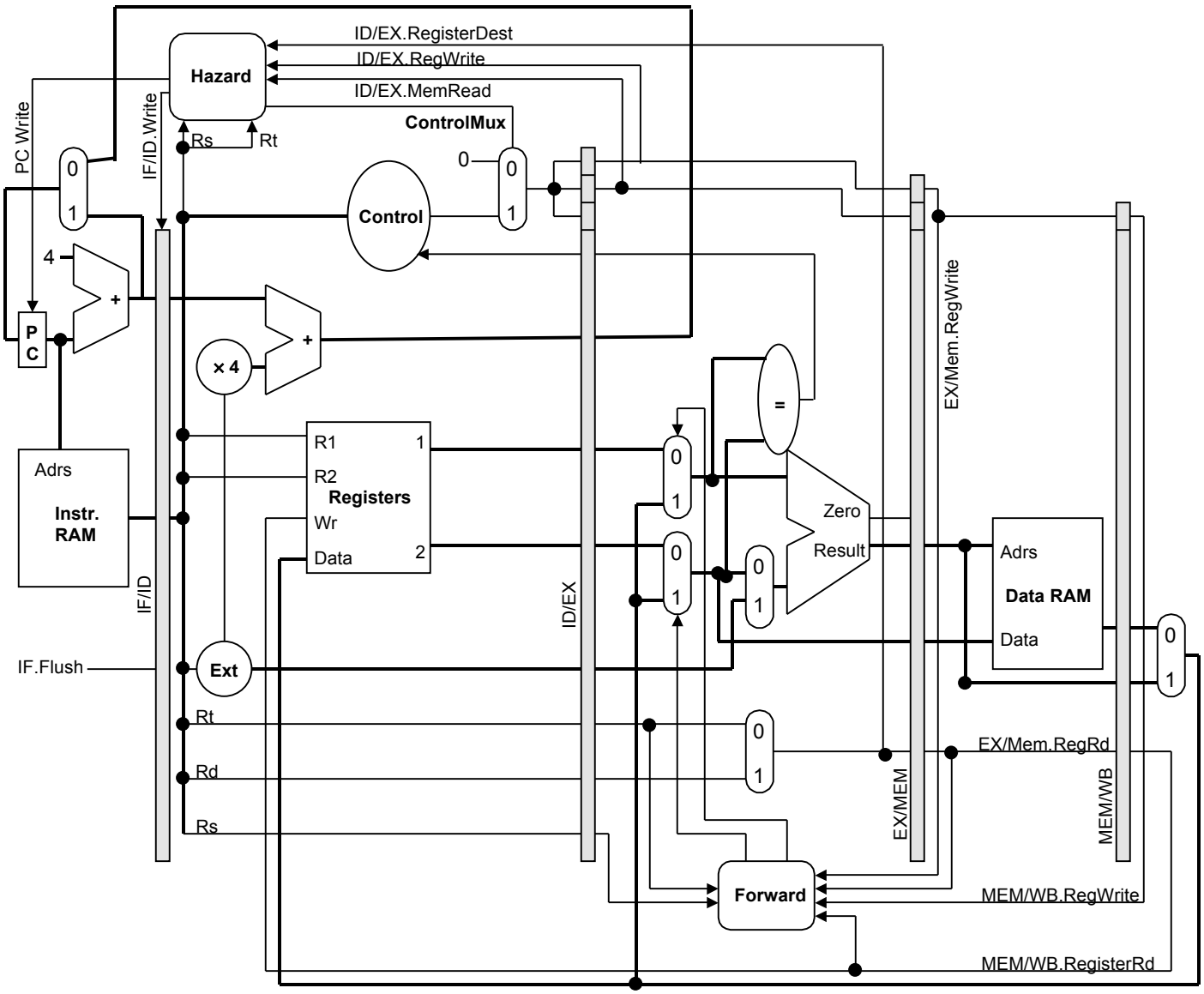
Note: While we're primarily concerned about correctness, full points will only be rewarded to solutions that do not lengthen the clock cycle. Assume that the ALU, Memory, and Register file all take 2ns, and everything else is instantaneous.

Part (b)

On the diagram below, write (next to the signal's name) values of **all** control signals required for the **sw+** instruction**.** (10 points)



Question 2: Pipelining (45 points)



Consider the data path shown above where only partial forwarding is implemented.

Part (a)

In what pipeline stage are branches resolved? How many cycles would we have to stall after a branch, if branches are not predicted? (5 points)

Branches are resolved in the EX stage. Since the correct instruction will be fetched when the branch is in the MEM stage, there would be 2 stall cycles.

Part (b)

Which forwarding paths are present? Give examples of instruction sequences that use **all** forwarding paths. (5 points)

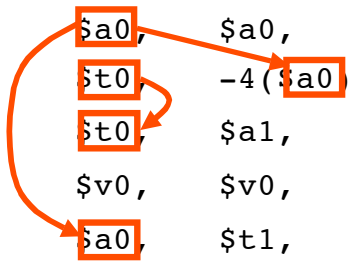
Forwarding from the WB stage (the MEM/WB latch) to both ALU inputs is provided. Both paths are used (without stalls) by the following code:

```
add     $t1, $t0, 1
nop
mul     $t2, $t1, $t1
```

Question 2, continued

The inner loop for the C code shown can be written as:

```
loop:   add     $a0,    $a0,    4
        lw      $t0,    -4($a0)
        bne     $t0,    $a1,    skip
        add     $v0,    $v0,    1
skip:   bne     $a0,    $t1,    loop
```



```
int count(int *A, int val, int len) {
  int count = 0, *end = A + len;
  do {
    if (*A == val) {
      count += 1;
    }
    A += 1; // pointer arith
  } while (A != end)
  return count;
}
```

Part (c)

Label all dependences within one iteration of this code (not just the ones that will require forwarding). *One iteration is defined as the instructions between the first* `add` *and the second* `bne`*, inclusive.* (5 points)

Part (d)

Using the pipeline shown on the previous page (assume branches are predicted **not taken**), determine in which cycle each instruction is in each pipeline stage using the grid below (stalls can be marked with an **S** or --). Assume that both branches are always taken. (20 points)

Inst	iter	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2
add	N	F	D	E	M	W																	
lw	N		F	D	-	E	M	W															
bne	N			F	-	D	-	E	M	W													
bne	N								F	D	E	M	W										
add	N+1											F	D	E	M	W							
lw	N+1												F	D	-	E	M	W					
bne	N+1													F	-	D	-	E	M	W			

Part (e)

Compute the average CPI for this loop. *Assume that the loop iterates indefinitely. It is okay to leave your answer as an expression.* (5 points)

10 cycles pass between the first fetch of the add and the second, so the time per iteration is 10 cycles. Thus,

CPI = 10 cycles/4 intructions = 2.5 cycles/instruction

Part (f)

If the pipelined machine has a clock cycle that is 4 times faster than a single cycle machine, what speedup is achieved from pipelining on this piece of code. *It is okay to leave your answer as an expression.* (5 points)

CPI(single-cycle) = 1; ClockPeriod(pipelined) = ClockPeriod(single-cycle)/4

Speedup(pipelined vs. single-cycle) = exec-time(single-cycle)/exec-time(pipelined)

= N * CPI(sc) * ClockPeriod(sc) / (N * CPI(pipe) * ClockPeriod(pipe))

= N * 1 * ClockPeriod(sc) / (N * 2.5 * ClockPeriod(sc)/4) = 1 / (2.5/4) = 4/2.5 = 1.6

Question 3: Concepts (24 points)

Write a short answer to the following questions. For full credit, answers should not be longer than **two sentences**.

Part a) Why is a special encoding necessary to represent the value zero in the IEEE floating point representation? (6 points)

In building the mantissa, an implicit one is added to the bits in the fraction field of the number, so there is no way to represent a zero .

Part b) What is the largest speedup that can be achieved by optimizing something that represents 20% of a program's execution time. *It is okay to leave your answer as an expression.* (6 points)

Using Amdahl's Law: new time = unimproved fraction + (improved fraction / local speedup)
(if local_speedup = infinity) = unimproved fraction

Speedup = old time / new time = 1 / unimproved fraction = 1/.8 = 1.25

Part c) When an interrupt occurs, the current PC is copied to an "exception PC" register. Why isn't it just written to register `$ra` as is done for `jal` instructions? (6 points)

The $ra register might be being used by the program. If we overwrite it with the exception PC, we will potentially break the program.

Part d) If a single-cycle machine has an instruction latency of 12ns and is pipelined into three stages, what can we say about its throughput? *Your answer should use proper units.* (6 points)

When we pipeline into 3 stages, we can reduce the clock period by _upto_ a factor of three (there is no guarantee that we'll get a factor of three). So the clock period could be as low as 4ns.

The pipelined machine can complete upto 1 instruction per cycle, so its

throughput <= 1 instruction/4ns