Midterm Exam

CSC 252
8 March 2018
Computer Science Department
University of Rochester

Instructor: Yuhao Zhu
TAs: Alan Beadle, Sayak Chakraborti, Michael Chavrimootoo, Alan Chiu,
Akshay Desai, Benjamin Nemeth, Eric Weiss, Jie Zhou

Name: Solution

Problem 0 (2 point): 2
Problem 1 (12 points): 12
Problem 2 (14 points): 14
Problem 3 (14 points): 14
Problem 4 (20 points): 20
Problem 5 (16 points): 16
Problem 6 (22 points): 22
Total (100 points): 100

Remember "I don't know" is given 15% partial credit, but you must erase/cross everything else.

Please be sure your name is on each sheet of the exam.

Your answers to all questions (and all supporting work) must be contained in the given boxes.

You have 75 minutes to work.

Please sign the following. I have not given nor received any unauthorized help on this exam.

Signature:______________________________

GOOD LUCK!!!

Problem 0: Warm-up (2 points).

Do you think Facebook will start making their own phones?

Any answer is accepted

Problem 1 (12 points. Suggested time: 10 mins)

Part a (3 points): Represent decimal value 66 in hexadecimal.

42

Part b (3 points): Represent hexadecimal value 0x11 in binary

10001

Part c (3 points): Octal is the base-8 number system, and uses the digits 0 to 7. Represent binary number 11110000 in octal.

360

Part d (3 points): Suppose we have two 16-bit 2's complement numbers:

01x1x0x1xx000xxx
100xxxx00xxx0x10

where some of the bits have not been identified, and they are represented by x.

Could the sum of these two numbers possibly result in an overflow? If yes, give an example. If no, please explain.

No. A positive number plus a negative number cannot overflow.

Problem 2 (14 points. Suggested time: 10 mins)

Part a (6 points): Express the two floating point numbers $2\frac{15}{16}$ and $\frac{1}{8}$ in binary normalized form.

(3 points) $2\frac{15}{16}$ in binary normalized form is:

1.0111×2^{1}

(3 points) $\frac{1}{8}$ in binary normalized form is:

1.000×2^{-3}

Part b (4 points): As you know IEEE floating point standard has a 32-bit representation and a 64-bit representation.

1	8	23
sign	exp	frac

(2 points) What is the bias used in the 32-bit representation?

127

$bias = 2^{8-1} - 1 = 127$

(2 points) What is the minimum gap between two representable subnormal numbers in the 32-bit representation?

$\frac{1}{2^{149}}$

$0.\underbrace{0 \cdots 1}_{23} \times 2^{0+1-127} = 2^{-149}$

Part c (4 points): In this problem, we assume that IEEE decided to add a new 10-bit representation, with its main characteristics consistent with the other two representations.

In this 10-bit representation, the value 33/256 is represented exactly as $0\underbrace{0011}_{4}\underbrace{00001}_{5}$. Your job: decide the number of bits needed for the exponent and for the fraction.

(2 points) Number of bits needed for exponent:

4

$$\frac{33}{256} = \frac{32}{256} + \frac{1}{256} = 2^{-3} + 2^{-8}$$
$$= 0.00100001$$
$$= 1.00001 \times 2^{-3}$$

(2 points) Number of bits needed for fraction:

5

Problem 3 (14 points. Suggested time: 5 mins)

Part a (6 points): Logic operations

(3 point) What is the result of bit-wise AND between 0101 and 1010?

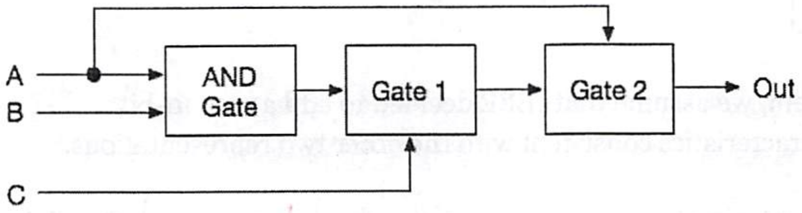
0000

(3 point) What is the result of bit-wise XOR between 0101 and 1010?

1111

Part b (8 points): The combinational circuit with three logic gates shown below takes in three 1-bit inputs: A, B, and C, and produces one 1-bit output: Out. The relationship between A, B, C, and Out is shown in the accompanying table, where the outputs for two of the input combinations are not specified (Out 1, Out 2). One of the logic gates in the circuit is already identified as an AND gate.

You job: complete the two missing gates in the combinational logic as well as the two missing outputs in the table. You can assume that the two missing gates will be either an AND gate or an OR gate.



A	B	C	Out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	Out 1
1	1	1	Out 2

(2 points) Gate 1:

OR

(2 points) Gate 2:

AND

(2 points) Out 1:

1

(2 points) Out 2:

1

Problem 4 (20 points. Suggested time: 15 mins)

Part a (6 points): Name three instructions that have zero operand, one operand, and two operands, respectively.

(2 points) zero-operand instruction:

ret

(2 points) one-operand instruction:

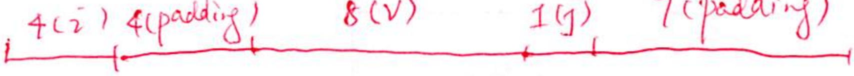
jmp

(2 points) two-operand instruction:

addq

Part b (5 points): Consider the following C declaration of an array of struct, assuming that the address of variable A is 0x8000 and that this program is executed on a 64-bit machine.

```
struct S {
   int i;       ← 4B
   double v;    ← 8B
   char j;      ← 1B
} A[10][10];
```



sizeof struct S = 8 × 3 = 24B

&A[2][3] = 0x8000 + (2×10+3)×24

&A[2][3].j = &A[2][3] + 16 = 0x8238

Essentially: (2×10+3)×24+16=568 = 0x238

What is the address of A[2][3].j?

0X8238

Part c (9 points): The following assembly program adds the absolute value of a signed integer stored in memory location 0x4000 to the absolute value of another signed integer stored in memory location 0x4004, and stores the sum to memory location 0x4008. Complete the three missing instructions.

Hint: This problem might test you on the following things:

- Register saving convention
- Control transfer
- Data movement

```
.func
  movq $0x4000, %rbx
  movq (%rbx), %rdi
  call .abs
  movq %rax, %rdx                (3 points)
  movq $0x4004, %rbx
  movq (%rbx), %rdi
  pushq %rdx                     (3 points)
  call .abs
  popq %rdx
  addq %rdx, %rax
  movq $0x4008, %rbx
  movq %rax, (%rbx)
  halt

.abs
  movq $0x0, %rdx
  addq %rdx, %rdi
  jns .done                      (3 points)
.skip
  negq %rdi
.done
  movq %rdi, %rax
  ret
```

Problem 5 (16 points. Suggested time: 15 mins)

Part a (4 points): (Fill in the blanks) Recall that the instruction cycle in the sequential implementation of a processor consists of six phases, not all of which are needed by all instructions. However, all instructions do need the

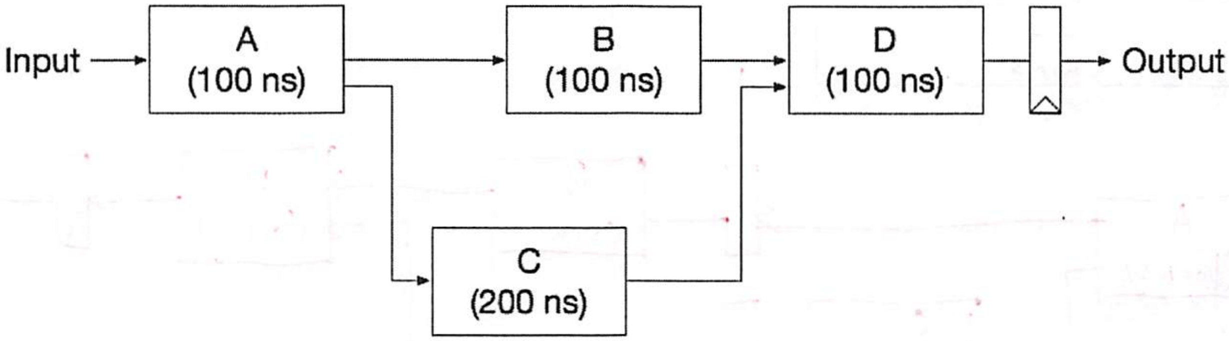
Fetch

stage and the

Decode

stage.

Part b (6 points): Consider the following combinational circuit, where the logic block A takes 100 ns, B takes 100 ns, C takes 200 ns, and D takes 100 ns; also assume that it takes 20 ns to load data into a register.



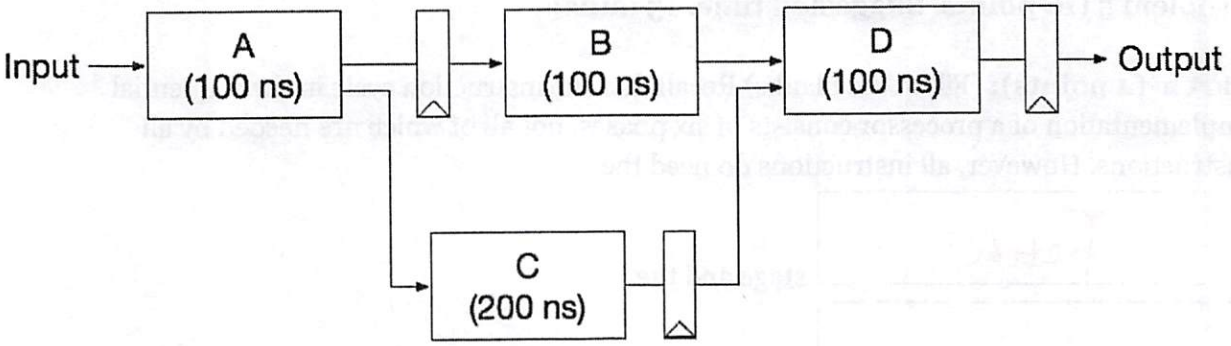
(3 points) What is the latency of this circuit from input to output?

$\overset{(A)}{100} + \overset{(C)}{200} + \overset{(D)}{100} + \overset{(Reg)}{20} = 420ns$

(3 points) What is the overall throughput of this circuit?

$\frac{1}{420ns}$

Part c (6 points): A student who just took CSC 252 came up with a pipelined implementation. She used two additional pipeline registers, and the new design is shown below. All the registers are connected to the clock signal, which is omitted in this figure for simplicity purpose.

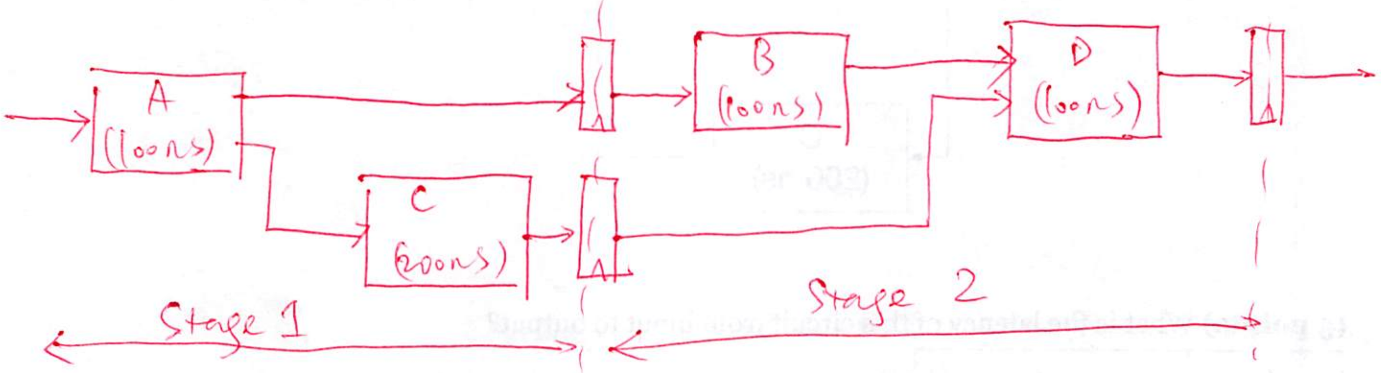


(3 points) What is the latency now?

640 ns

(3 points) What is the overall throughput now?

$\frac{1}{320\text{ns}}$



① Stage 1 Latency = 100 + 200 + 20 = 320 ns

② Stage 2 Latency = 100 + 100 + 20 = 220 ns

③ 320 > 220

④ So cycle time is determined by Stage 1 (320 ns)

⑤ Cycle time = 320 ns

⑥ Latency = 320 × 2 = 640 ns

⑦ Throughput = $\frac{1}{320\text{ns}}$. That is, a new instruction can be pushed to the pipeline every 320 ns

Problem 6 (22 points. Suggested time: 20 mins)

Consider the following assembly code (.L1 is a label. I1 ~ I7 are just notations we add so that we can easily refer to an instruction; they are not part of the program):

```
I1:        movq $0, %rax
        .L1:
I2:         movq %rdi, %rdx
I3:         andq $1, %rdx
I4:         addq %rdx, %rax
I5:         shrq $1, %rdi
I6:         jne .L1
I7:         ret
```

x4

Popcount

page 82 in Lecture7.pdf

Assume that %rdi is loaded with 0x0F before the program starts.

Part a (3 points): What is the value of %rax after the code returns?

4

Part b (3 points): In a single-cycle, sequential microarchitecture, how many cycles does this program take to finish?

22

1+4×5+1

Part c (9 points): There are 4 pairs of read-after-write data dependencies. One of them is <I1, I4> because I1 writes to %rax and I4 reads from %rax. What are the other three pairs? Write the answers in the form of <Im, In>.

(3 points) Pair 1

<I2, I3>

(3 points): Pair 2

<I3, I4>

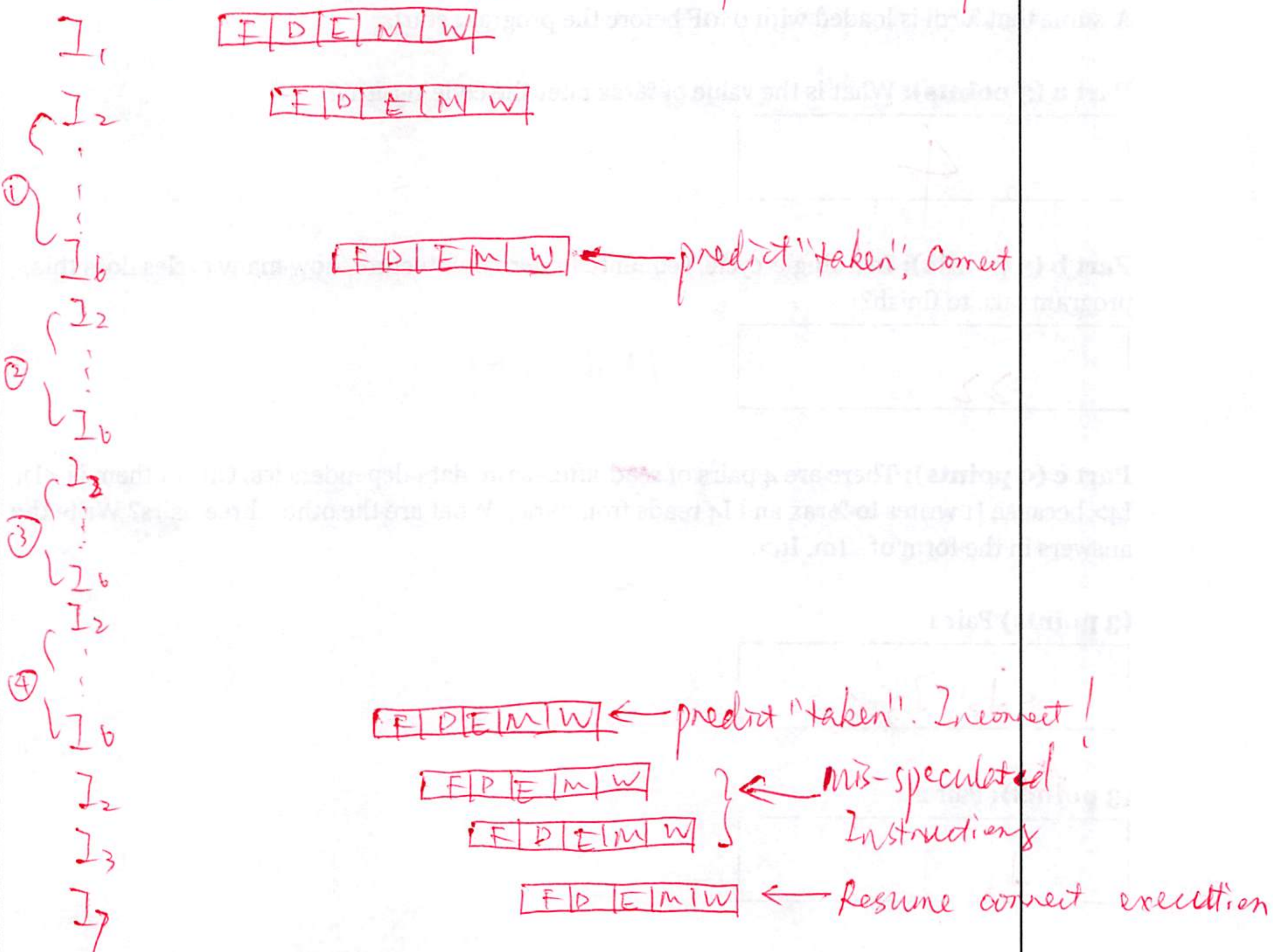
(3 points): Pair 3

<I5, I2>

Part d (7 points): One student implemented a 5-stage pipeline processor as we described in the class as well as in the textbook. All data dependencies can be handled using data forwarding in her design (i.e., no pipeline bubble due to data dependency). Also, she implemented a 1-bit branch predictor as we discussed in the class, which always uses the last prediction result and changes mind instantaneously after a misprediction. You can assume that the first prediction is made as "taken."

You job: determine how many cycles this particular processor takes to finish the above program. Show your work to obtain partial credit.

* Only control dependency stalls the pipeline in this problem



Total cycles = 1 + 5 × 4 + 2 + 5 = 28 cycles