CISS360: Computer Systems and Assembly Language
Final f01

Name: jdoe5@cougars.ccis.edu Score: □

Open `main.tex` and enter answers (look for `answercode`, `answerbox`, `answerlong`). Turn the page for detailed instructions. To rebuild and view pdf, in bash shell execute `make`. To build a gzip-tar file, in bash shell execute `make s` and you'll get `submit.tar.gz`.

FALL2023 NEW INSTRUCTIONS: The first bullet point below is changed to the following – This is open-book and takehome and no discussion with anyone (but yourself). You can only use my notes. Sure you can use a calculator, but you have to justify all calculations anyway.

Instructions

- This is a closed-book, no-discussion, no-calculator, no-browsing-on-the-web no-compiler/no-MIPS-simulator test.
- Cheating is a serious academic offense. If caught you will receive an immediate score of -100%.
- If a question asks for a program output and the program or code fragment contains an error, write `ERROR` as output. When writing output, whitespace is significant.
- If a question asks for the computation of a value and the program or code fragment contains an error, write `ERROR` as value.

Honor Statement

I, [REPLACE WITH YOUR FULLNAME], attest to the fact that the submitted work is my own and is not the result of plagiarism. Furthermore, I have not aided another student in the act of plagiarism.

```
IMPORTANT INSTRUCTION FROM EARTH:
SHOW AND EXPLAIN ***ALL*** YOUR WORK.
YOU ARE NOT ALLOWED TO SHARE THIS DOCUMENT AFTER THE EXAM.
END OF TRANSMISSION FROM HOUSON.
```

Question	Points
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	

Question	Points
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	
31	
32	
33	
34	
35	
36	
37	
38	
39	
40	

TOTAL	

Welcome to your first day of work at Fullabugz Company. Your job is to write MIPS code, debug MIPS code (written by someone else ... ARGHHH), etc.

Do not give me two answers for the same question. I reserve the right to choose one for you ... and I always choose the wrong one (if there is one)!

Here's some useful info:

1. Print int: syscall 1
2. Read int: syscall 5
3. Print string: syscall 4
4. Read string: syscall 8
5. Exit: syscall 10

You should know what to do with `$v0`, `$a0`, `$a1`, right?

Here's a table of powers of 2:

n	2^n
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096
13	8192
14	16384
15	32768
16	65536
17	131072
18	262144
19	524288
20	1048576

Q1. The saga of an interplanetary software consultant continues ...

The fee of your consulting work for planet Tutu is $51.765. But they only accept your bill in base 2. Convert 51.765 to base 2 up to 3 binary place (but with possible rounding from the 4th binary place.) Show all work.

Answer: In the following use _ for subscript and ^ for power.

```
First we consider 51.

   2 | 51
     +-------
   2 | ?? r ?
     +-------

Therefore 51_{10} = ??????_2

Now we consider 0.765_{10}.

      2 x 0.765 = ?.???
      2 x 0.??? = ?.???

Therefore 0.765_{10} is approxmately 0.????_{2} up to 4 binary place.
Therefore up to 3 binary places (with rounding), 0.765_{10} = 0.???_{2}.

Therefore 51.765_{10} = ?.?_{2}.
```

Q2. Something seems to wrong with the mainframe computer on planet Rouf. Their machine is basically in base 4. After tracing some output, you suspect something is wrong with the base 4 multiplication in the chip. You decide to double check at least one case by hand ...

(a) Compute the following product in base 4:

$$120021_4 \times 103011_4$$

(b) Convert the result of (a) to base 10.

(c) Convert 120021_4 to base 2 quickly.

(d) Convert 103011_4 to base 2 quickly.

Answer: In the following use _ for subscript and ^ for power.

```
(a)
                    120021
                  x 103011
           ---------------





           +
           ---------------

           ---------------

(b) ?????_4 = ? x ? + ? x ? + ... (complete this expression using ^ for power)
            = ? + ? + ...
            = ?

(c) 120021_4 = (1|2|0|0|2|1)_4
             = (  |  |  |  |  |  )_2
             = ?_2

(d) 103011_4 =
```

Q3. While traveling back to Earth, you were abducted by the Glinkons: unfortunately your teleporter broke down and in an attempt to get home in time for the beginning of the football season, you thought it was safe to take a shortcut through their spacial territory ...

When their commander accused of spying on their planet, you blurted out that you're just a software engineer ... a geek ...

"Oh a CS geek, eh? Really?!? In that case you won't have trouble solving this problem ..."

"The number in 1011001 in base 2 has three 0s and a total length of 7. So the ratio of 0s is 3/7. Right? Of course left trailing zeroes do not count."

(You're sweating now ... it's one of those pesky number base problems again ...)

"But this same number 1011001 in base 2 when converted to 10 is 89 which has no zeros. So the ratio of 0s is 0/2."

"So I'm giving you the number 49618850 in base 10. Find the base B such that when this number is converted to this base B, the representation has the largest ratio of zeros. B is at most 16."

"You have 1 hour. SHOW ALL YOUR WORK!!!"

Answer:

```
base B    representation in base B       length   number of zeroes ratio
------    -----------------------------  ------   ---------------- -----
     2
     3
     4

    16

Therefore the base with the largest ratio of zeroes is B = ?.
```

Q4. (a) Many of the computers at planet Ducloz Dusun are all fried because of solar flares ... and you're on that planet right now to help them. Fortunately many computers shutdown in time. To bring up all the functioning hardware, you need to key in the sys admin password, which is the binary sequence of the two's complement representation of -96 on their mainframe. (They can't do it because they have forgotten their binary math.) Note that their word size is 35 bits, i.e. registers are made up of 35 bits.

(b) The people Ducloz Dusun felt dumb, not able to restart their machines because they forgot their binary math for 5000 years. They wanted to learn their math all over again and your boss happily obliged because he's going to charge them 5x the usual overtime pay. Unfortunately you have to teach them the computer math. In particular, you have to show them how to perform

$$42 - 13$$

on their 35 bit machine but using only addition.

Answer:

```
Write down 42 and -13 in base 2 for 35-bit wide registers.
Perform 35-bit wide addition. Convert that back to base 10.

    42 = ?_2

    13 = ?_2

Flipping the bits of 13, we get:

    ?_2

Adding a 1, we get -13:

    ?_2

Therefore 42 - 13 = 42 + (-32) is


        +
        -------------------------------------------------

        -------------------------------------------------
```

Q5. Great ... your boss bought a nanophone, a really tiny mobile phone. MIPS is already a RISC machine, but your manager's nanophone is a reduced MIPS. (RRISC?) It particular, it does not have arithmetic instructions like add, addi, addiu, sub, etc. (This is definitely not a smartphone ...)

Your manager asked you to write a function to perform unsigned addition of two registers. You checked the technical documentation for this reduced MIPS chip and see that although it does not have arithmetic instructions, it does have all other instructions and pseudoinstructions. It also contains logical operations (i.e. bit operations). For instance:

```
                        # Description               Description using C/C++
sll   $s0, $s1, 2       # shift left by constant    s0 = s1 << 2
srl   $s0, $s1, 3       # shift right by constant   s0 = s1 >> 3
sllv  $s0, $s1, $t0     # shift left by variable    s0 = s1 << t0
srlv  $s0, $s1, $t1     # shift right by variable   s0 = s1 >> t1

and   $s0, $s1, $s2     # and                       s0 = s1 & s2
andi  $s0, $s1, 4       # and immediate             s0 = s1 & 5

or    $s0, $s1, $s2     # or                        s0 = s1 | s2
ori   $s0, $s1, 5       # or immediate              s0 = s1 | 5

nor   $s0, $s1, $s2     # not-or (nor)              s0 = ~(s1 | s2)
```

(See chapter #25 of my notes.) You should call your function `myaddu`. Of course it performs unsigned add on `$a0`, `$a1` and puts the resulting bits in `$v0`. You should add test code as given below.

Your manager is waiting for your work and will be testing it on PCSpim before using the code in his nanophone. He's also going to get Joe Cantcode to visually verify that your function does NOT contain any arithmetic instructions (i.e., no addition, no subtraction, no multiplication, no division, whether signed or unsigned.)

Answer: Copy and paste your code below. I must be able to run it.

```
        .text
        .globl main

myaddu:
        # TODO ...
        # Perform unsigned add on $a0, $a1 and store result in $v0

main:                           # perform syscall to read int x and y and
                                # store them in $a0, $a1
        jal     myaddu          # compute the unsigned sum of $a0, $a1
                                # and store result in $v0

                                # perform syscall to print int the return
                                # value in $v0 which is set by myaddu

        li      $v0, 10
        syscall
```

Q6. Great. Just great. Someone at Fullabugz accidentally deleted the source code for one of their most important program ... the payroll program!!! Everyone is given some MIPS machine code to disassemble, i.e., to translate into MIPS assembly code. Here's yours machine code in hexadecimal:

`0x01099022`

Your boss told you that the above machine code is an R-format instruction. You quickly google for MIPS machine instruction format and quickly look for R-format instructions ...

(a) First the manual tells you to translate the above instruction `0x01099022` in hexadecimal into 32 bits.

(b) The manual tells you that the 32 bits from (a) is broken up into fields. Going left to right, the first 6 bits is called `op`, the next 5 bits is called "rs", and next 5 bits is called "rt", the next 5 bits is called "rd", the next 5 bits is called "shamt", and the remaining rightmost 6 bits is called "funct". Write down the fields in base 2.

(c) In base 10, write down the op, shamt, and funct for the machine instruction `0x01099022`.

(d) The manual tells that rs, rt, rd are translated to registers by translating the base 2 numbers rs, rt, rd to base 10. For instance if the binary value of rt is 8, then it refers to register `$8`, i.e., it refers to the register name `$t0`. Write down the registers rs, rt, rd in base 10 and then their names.

(e) Finally, the manual tells us that for an R-format machine instruction is translated to the following assembly instruction:

```
    [operation]    [rd], [rs], [rt]
```

Answer:

```
(a) The machine code in 32 bits (for readability, group bits in chunks of 4
bits):

        0x01099022 = ???? ???? ???? ???? ???? ???? ???? ????

(b) We break up the bits from (a) into chunks of 6 bits, 5 bits, 5 bits,
5 bits, 5 bits, 6 bits to get the op, rs, rt, rd, shamt, funct fields:

        ??????  ?????  ?????  ?????  ?????  ??????

The fields of the machine code 0x01099022 is
```

	in base 2	in base 10
op	?	?
rs	?	?
rt	?	?
rd	?	?
shamt	?	?
funct	?	?

(c) The operation is ?.

(d) For our machine instruction the register names of the rs, rt, rd are:

	register name
rs	?
rt	?
rd	?

(e) Finally the assembly instruction for the machine code 0x01099022 is

? ?, ?, ?

Q7. The CEO has decreed that your pay is going to be `f(10)` where `f` is the function:

```
int f(int n)
{
    if (n == 0) return 5;
    else if (n == 1) return 7;
    else if (n == 2) return 3;
    else return 2 * f(n - 1) + 3 * f(n - 2) + 4 * f(n - 3);
}
```

but only if you can implement the above in MIPS. What kind of a crazy company is this?!?

Answer: Copy and paste your code below. I must be able to run it as a complete program.

```
        .text
        .globl main

f:      # TODO


main:                           # perform syscall to read int n and
                                # store them in $a0

        jal     f               # compute f(a0) and store result in $v0

                                # perform syscall to print int the return
                                # value in $v0 which is set by f

        li      $v0, 10
        syscall
```

Note: Of course you are advised to test your code. As always, if your program crashes or runs an infinite loop during a test run, you get 0 for that test case.

Q8. Your work on Planet Taolf, their civilization now has advanced to the point where they now need more than just integers! Time for floating point numbers! This curious civilization manage to move to computers with only the concept of integers. You have decided not to contact IEEE to design a new floating point type – you are going to design it yourself (because you don't want to outsource and you want to keep all the consulting fees). Well, not exactly, you are going to copy their design. There's just one problem: their computers uses 30 bits registers and not 32. You have decided to follow the IEEE 754 Single Precision representation as much as possible. The 30-bit computers on Toalf will be laid out with 3 fields just like IEEE 754 Single Precision format:

- 1 bit for the sign bit s.
- 8 bits for encoding the exponent E. The bias is the same, i.e., 127.
- 21 bits for the fractional part F of the mantissa.

You are calling this the TAOLF30 standard.

(a) You are going to test your Toalf computer with TOALF30 floating point. Here's your test case: What is the 30-bit number `011011010010101011111000001001` written in TAOLF30 in our usual normalized base 10 scientific notation?

(b) Now you need to bill Toalf. It's going to be $57,130,500.55. But you are now on Earth. So you need to convert 57130500.55 into 30 bits and send the bits to Toalf by intergalactic-network. What is the 30-bit representation of 57130500.55 using the TAOLF30 representation?

Answer:

```
(a) The fields of 011011010010101011111000001001 are

    s =
    E + Bias =
    F =

Therefore

    sign = ? (put + or -)
    E =
    1 + F =

Converting to base 10

    sign = ? (put + or -)
    E =
    1 + F =

Hence in signed fractional binary form, 011011010010101011111000001001 is
```

```
    (+ or -) 1.? x 2^? =

Converting the above to normalized base 10 scientific notation, we get

    ? = ?

(b)
```

Q9. You are told to design the following for a game show on planet Nemow. The design will tell the game host who which of two teams will get to answer the question.

There are two husband-and-wife teams. Let's call the input from the husband of the first team w and the input his wife x. The input from the husband of the second is y and the input from his wife is z. An output of 0 mean team 1 will answer and an output of 1 will mean team 2 gets to answer.

On planet Nemow, during election, each woman's vote count as 2 points and a man's vote count as 1. The game show host has decided to use the same rule for his show. After reading a question, members can press a button to indicate interest in answering the question. If the husband from team 1 presses his button, which for us will mean $w = 1$, and his wife also presses her button, i.e., $x = 1$, they get 3 points. Suppose for team 2, the husband does not press his button, i.e., y = 0, but his wife presses her button, i.e., $z = 1$. So team 2 gets 2 points. Your machine will decide based on which team has more "button points". In this case, since team 1 has 3 points and team 2 has 2 points, team 1 gets to answer. Your machine will output $f = 0$. An output of $f = 1$ means team 2 gets to answer. For instance suppose for team 1, the husband presses the button and his wife does not (team 1 has 1 button point), and for team 2, the husband does not press his button but his wife does (team 2 has 2 button points). Therefore in this case team 2 gets to answer; your device will output $f = 1$.

In the case of a tie breaker, team 1 gets to answer. That sounds like a good deal. However, if no one wants to answer, team 1 has to answer!

Fortunately you only need to provide a logic design. Make sure you work out each part carefully and correctly. I will stop reading your work once you hit a mistake.

(a) Write down the truth table for f.
(b) Write down the K-map for f.
(c) Derive a minimal boolean expression for f (in SOP form).
(d) Draw the logic network diagram for f. (See below for example.)
(e) Write down the cost of the logic network. (This can be calculated this way: Ignoring the NOT gates, count the number of inputs to all the AND and the OR gates.)

Answer:

```
(a)     +---------+---+
        | w x y z | f |
        +---------+---+
        | 0 0 0 0 | ? |
        | 0 0 0 1 |   |
        | 0 0 1 0 |   |
        | 0 0 1 1 |   |
        | 0 1 0 0 |   |
        | 0 1 0 1 |   |
        | 0 1 1 0 |   |
        | 0 1 1 1 |   |
        | 1 0 0 0 |   |
        | 1 0 0 1 |   |
        | 1 0 1 0 |   |
        | 1 0 1 1 |   |
        | 1 1 0 0 |   |
        | 1 1 0 1 |   |
        | 1 1 1 0 |   |
        | 1 1 1 1 |   |
        +---------+---+

(b)   yz   00  01  10  11
   wx    +----------------+
      00 |  ?   ?   ?   ? |
      01 |                |
      11 |                |
      10 |                |
        +----------------+

(c) f =

(d)

(e) Cost =
```

Here's an example of how to draw logic networks using ASCII art:

```
                             +----+ ab
      a ---+---------------|AND |-----+
           |           +---|    |     |   +----+ ab + a'b
           +--NOT --+  |   +----+     +---|OR  |-----------
                    |  |              +---|    |
                    |  |   +----+     |   +----+
                    +--|---|AND |-----+
      b ---------------+---|    | a'b
                           +----+
```

The above logic network has a cost of 2 + 2 + 2 = 6.

Instructions

In `main.tex` change the email address in

```
\renewcommand\AUTHOR{jdoe5@cougars.ccis.edu}
```

yours. In the bash shell, execute "`make`" to recompile `main.pdf`. Execute "`make v`" to view `main.pdf`. Execute "`make s`" to create `submit.tar.gz` for submission.

For each question, you'll see boxes for you to fill. You write your answers in `main.tex` file. For small boxes, if you see

```
1 + 1 = \answerbox{}.
```

you do this:

```
1 + 1 = \answerbox{2}.
```

`answerbox` will also appear in "true/false" and "multiple-choice" questions.

For longer answers that needs typewriter font, if you see

```
Write a C++ statement that declares an integer variable name x.
\begin{answercode}
\end{answercode}
```

you do this:

```
Write a C++ statement that declares an integer variable name x.
\begin{answercode}
int x;
\end{answercode}
```

`answercode` will appear in questions asking for code, algorithm, and program output. In this case, indentation and spacing is significant. For program output, I do look at spaces and newlines.

For long answers (not in typewriter font) if you see

```
What is the color of the sky?
\begin{answerlong}
\end{answerlong}
```

you can write

```
What is the color of the sky?
\begin{answerlong}
The color of the sky is blue.
\end{answerlong}
```

For students beyond 245: You can put LaTeX commands in `answerlong`.

A question that begins with "T or F or M" requires you to identify whether it is true or false, or meaningless. "Meaningless" means something's wrong with the statement and it is not well-defined. Something like "$1+_2$" or "$\{2\}^{\{3\}}$" is not well-defined. Therefore a question such as "Is $42 = 1+_2$ true or false?" or "Is $42 = \{2\}^{\{3\}}$ true or false?" does not make sense. "Is $P(42) = \{42\}$ true or false?" is meaningless because $P(X)$ is only defined if X is a set. For "Is 1 + 2 + 3 true or false?", "1 + 2 + 3" is well–defined but as a "numerical expression", not as a "proposition", i.e., it cannot be true or false. Therefore "Is 1 + 2 + 3 true or false?" is also not a well-defined question.

When writing results of computations, make sure it's simplified. For instance write 2 instead of $1 + 1$. When you write down sets, if the answer is $\{1\}$, I do not want to see $\{1, 1\}$.

When writing a counterexample, always write the simplest.

Here are some examples (see `instructions.tex` for details):

1. T or F or M: 1 + 1 = 2 .... $\boxed{\text{T}}$
2. T or F or M: 1 + 1 = 3 .... $\boxed{\text{F}}$
3. T or F or M: $1+^2 =$ .... $\boxed{\text{M}}$
4. $1 + 2 = \boxed{3}$
5. Write a C++ statement to declare an integer variable named `x`.

   ```
   int x;
   ```

6. Solve $x^2 - 1 = 0$.

   Since $x^2 - 1 = (x - 1)(x + 1)$, $x^2 - 1 = 0$ implies $(x - 1)(x + 1) = 0$. Therefore $x - 1 = 0$ or $x = -1$. Hence $x = 1$ or $x = -1$.

7. Which is true? .... $\boxed{\text{C}}$

   (A) $1 + 1 = 0$
   (B) $1 + 1 = 1$
   (C) $1 + 1 = 2$
   (D) $1 + 1 = 3$
   (E) $1 + 1 = 4$