ZHANG | | CHUE
Computer Science | or | ***Computer Engineering***

Computer Science
C.Sc. 342

YOU CAN STUDY FOR THIS TEST ANY TIME UNTIL SEPTEMBER 20, 2021, 12:00 PM

TAKE HOME Prereq TEST will be graded!

TEST TIME 12:00-1:40PM, and 5:00PM - 6:15 PM September 20, 2021

PLEASE MARK YOUR Major: CPE or CSc

Did you take CS 211 Date: FALL 2019

Did you take EE 210 Date : Cs 210 Date:

NONE of the Above

Please hand write and sign statements affirming that YOU WILL NOT CHEAT:

"I will neither give nor receive unauthorized assistance on this exam. I will use only one computing device to perform this test"

Please sign:

I will neither give nor receive unauthorized assistance on this exam. I will use only one computing device to perform this test

NO CORRECTIONS ARE ALLOWED IN YOUR ANSWERS!!!!!

You may use the back page for computations.
Please answer all questions.

PLEASE WRITE SHORT EXPLANATION ON HOW DID YOU ARRIVE AT THE ANSWER. Just giving final answer will result in ZERO grade for the question.

ZHANG
Computer Science or CHUE
Computer Engineering

Part A. Number Representation Systems

Please select all **TRUE** Boolean expressions (Note: the left byte is the most significant, signed numbers use two's complement representation):

Question A_1 (10 points)

signed signed
`0x1001>0x8001` , `0x1001<0x8001`,

0x8001 = 1000 0000 0000 0001= **binary**
complement = 0111 1111 1111 1111
decimal = -32676

0x1001 = 0001 0000 0000 0001= **binary**
complement = 1110 1111 1111 1111
decimal = 4097

Question A_2 (10 points)

unsigned unsigned
`0x1001>0x8001`, `0x1001<0x8001`

0x8001 = 1000 0000 0000 0001 = **binary**
decimal = 32769

0x1001 = 0001 0000 0000 0001= **binary**
decimal = 4097

Question A_3 (10 points)

signed signed
`0x8FFF>0x7FFF` , `0x8FFF<0x7FFF`,

0x8FFF = 1000 1111 1111 1111 = **binary**
complement = 0111 0000 0000 0001
decimal = -28673

0x7FFF = 0111 1111 1111 1111 = **binary**
complement = 1000 0000 0000 0001
decimal = 32767

Question A_4 (10 points)

unsigned unsigned
`0xF000>0x7FFF`, `0xF000<0x7FFF`

0xF000 = 1111 0000 0000 0000 = **binary**
decimal = 61440

0x7FFF = 0111 1111 1111 1111 = **binary**
decimal = 32767

ZHANG
Computer Science or CHUE
Computer Engineering

Question A_5 (10 points)

`signed` `signed`
`0xF000>0x7FFF,` `0xF000<0x7FFF`

0xF000 = 1111 0000 0000 0000 = **binary**
complement = 0001 0000 0000 0000
decimal = -4,096

0x7FFF = 0111 1111 1111 1111 = **binary**
complement = 1000 0000 0000 0001
decimal = 32767

Question A_6 (10 points)

signed signed
`0x9000>0x7FFF ,` `0x9000<0x7FFF`

0x9000 = 1001 0000 0000 0000 = **binary**
complement = 0111 0000 0000 0000
decimal = -28672

0x7FFF = 0111 1111 1111 1111 = **binary**
complement = 1000 0000 0000 0001
decimal = 32767

Question A_7 (10 points)

`unsigned` `unsigned`
`0x8000>0x7FFF` `0x8000<0x7FFF`

0x8000 = 1000 0000 0000 0000 = **binary**
decimal = -32768

0x7FFF = 0111 1111 1111 1111 = **binary**
decimal = 32767

Question A_8 (15 points)

a. Convert two's complement signed 8 bit binary integer **`11111110`**

- to decimal number
- to hexadecimal representations
- to Octal representation

Decimal = (0 x 2^0) + (1 x 2^1) + (1 x 2^2) + (1 x 2^3) + (1 x 2^4) + (1 x 2^5) + (1 x 2^6) - (1 x 2^7) = -2

Hexadecimal = F = 1111, E = 1110 therefore 0xFE, there is a table for this... 0 = 0000, 1 = 0001... F = 1111

Octal = 11111110 = (011) (111) (110) = 376, I broke them into 3-bit integers so they can be octal

ZHANG *Computer Science*	or	CHUE *Computer Engineering*

Question A_9 (15 points)

Convert **1023** in decimal

b. to binary (unsigned)
c. to hexadecimal.

b.	c.
1023/2 = 511 r 1 511/2 = 255 r 1 255/2 = 127 r 1 127/2 = 63 r 1 63/2 = 31 r 1 31/2 = 15 r 1 15/2 = 7 r 1 7/2 = 3 r 1 3/2 = 1 r 1 1/2 = 0 r 1 **binary(unsigned)** = 1111111111	**binary(unsigned)** = 0011 1111 1111 0011 = 3 1111 = F Therefore, answer is **Hexadecimal** = 0x3FF

Question A_10 (15 points) 32 bit float LOAT (FLOATING POINT) Representation of -0.75

Sign	**Exponent**	**Mantissa**
	0.75 x 2 = 1.5 | 1 0.50 x 2 = 1.0 | 1 **Binary** = 0.11 0.10 x 2^-1 -1 + 127 = 126 **exponent** = 126 126 = 0111 1110	0.10 = **decimal** Therefore, mantissa is 1000 0000 0000 0000 0000 0000
1	0111 1110	1000 0000 0000 0000 0000 0000

Question A_11 (15 points) 32 bit Fixed POINT Representation of -0.75

sign = 1, -0.75 is negative
.75 x 2 = 1.5 | 1
0.5 x 2 = 1.0 | 1
binary of decimal = 11
binary of integer = 0
therefore

sign	**integer**	**fraction**
1	0000 0000	1100 0000 0000 0000 0000 0000

115/130

ZHANG CHUE

Computer Science or ***Computer Engineering***

Part B. Programing

Question B_1 (15 points)
Please define (describe) local ***variables*** and how they are used in computer programming (e.g. In C, C++).
Please write clearly and use no more than 2 sentences.

Local variables are variables that are defined and used for a specific part of the program, for instance, a function and that variable could only be used in that function but can be returned. Often times we see local variables being defined then used as an argument in functions

Question B_2 (15 points)
Please define (describe) ***static variables*** and how they are used in computer programming (e.g. In C, C++). Please write clearly and use no more than 2 sentences.

Static variables are variables that can maintain their state globally while still being in a function, for example, suppose we define a static variable within a function that increments per function call and call it multiple times. If it were a local variable, the variable would reset and start at 0 every time of the function call but for static variables, the variable would not reset and would be N, the number of times the function was called.

Question B_3 (15 points)
Please define (describe) ***const variables*** and how they are used in computer programming (e.g. In C, C++). Please write clearly and use no more than 2 sentences.

Constant variables are variables that cannot be modified once defined, for example, a constant variable defined cannot be incremented and must stay as the same value. This is often used to ensure that a variable isn't changed on accident.

Question B_4 (15 points)
Please explain the differences and similarities between ASCII and binary integer representations. Please write clearly and use no more than 2 sentences.

The similarities between ASCII and binary inteteger representation is that data is often converted to binary or ASCII and ASCII and binary both use bits, but ASCII uses 7 bits only while binary uses any amount.

ZHANG		CHUE

ZHANG		CHUE
Computer Science	or	***Computer Engineering***

Question B_5(20 points)

Convert this function into pointer-based code using C language.

```
void shift(int a[], int n) {
   int i;
    for(i = 0; i != n-1; i++)
      a[i] = a[i+1];
  }
```

```
void shift(int a[], int n) {
      int i, *ptr[n];
      for(i = ; i < n-1; i++)

              ptr[i] = &a[i+1];
          }

Void shift(int *a, int n){
     Int I;
     for(I = 0; I < n-1; I++){
            *(a+I) = *(a + I + 1);
         }
}
```

Here are two variants to the pointer-based code using C language. The top variant has a pointer variable ptr which in which we reference the array A and then shift the value.

The bottom variant showcases an array pointer as the parameter and we modify the array by calling the nth index of the array, then shift the value

ZHANG | | CHUE
Computer Science | or | *Computer Engineering*

Question B_6(20 points)

Complete the following `setName`, `getStudentID`, and `setStudentID` functions. You may assume the pointers given are valid and not null.

```
        #define MAX_NAME_LEN 127

      typedef struct {
            char name[MAX_NAME_LEN + 1];
             unsigned long sid;
        } Student;

      /* return the name of student s */
const char* getName (const Student* s) {
               return s->name;
        }

        /* set the name of student s
 If name is too long, cut off characters after the maximum number of
characters allowed.
        */
        void setName(Student* s, const char* name) {

                 int namelen = strlen(namelen)
                 int i = 0
                 while(namelen < MAX_NAME_LEN){
                      s->name[i] = name[i]
                      i++
                             }
        }
```

Because we are using C and there is there is a need to iterate through any types of arrays, I used a while loop to check if the length of the name is less than the max name length and until it is reached. This way, the excess name will be truncated.

```
        /* return the SID of student s */
        unsigned long getStudentID(const Student* s) {
                 return s->sid;
        }
```

Standard method for returning

```
/* set the SID of student s */
        void setStudentID(Student* s, unsigned long sid) {
                  s->sid = sid
        }
```

Standard method for setting

70/120

ZHANG | CHUE
Computer Science or *Computer Engineering*

Part C. Digital Logic

Question C_1(20 points)
Please describe the differences and similarities between Latches and FlipFlops.
Please write clearly and use no more than 2 sentences.

Latches and FlipFlops both take 2 inputs and outputs 1 where if one suppose for inputs S and R, S = 0 and R = 1, then output will be either active high or active low, Q and Q*. The difference between Latches and FlipFlop comes from the fact that Latches can have input of both S = 0 and R = 0 but can't have S = 1 and R = 1 while the opposite logic is maintained by FlipFlop cirtuicts and this is because Latches use NOR gates and FlipFlop uses NAND gates.

Question C_2(20 points)
Please write BOOLEAN function for 2:1 multiplexer

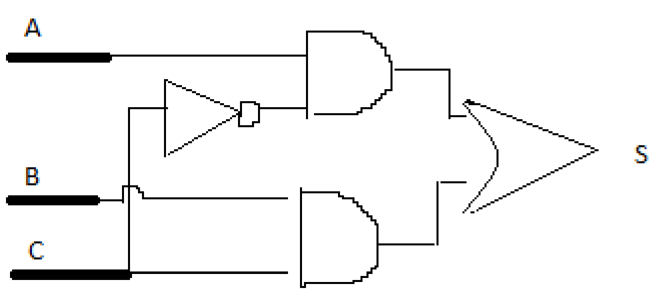
A = input 1
B = input 2
C = Cin
S = Output

Boolean Function
==============
A(C*) + B(C) = S

Question C_3(20 points)
Please draw a TRUTH table for 2:1 multiplexer
A is input 1,
B is input 2,
C is Cin or the Selector,
S is the Output

A	B	C	S
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	1

Question C_4(20 points)
Please draw schematic diagram using AND, OR, NOT gates for 2:1 multiplexer



60/80

ZHANG
Computer Science or CHUE *Computer Engineering*

Question C_5(20 points)
Please describe SETUP and HOLD times in digital circuits.

SETUP time is basically the time needed before an active edge is triggered and the HOLD time is basically the time needed after an active edge is triggered. The SETUP and HOLD timers are the minimum time required for the state to be held at a constant after input and output (respective).

Part D. Assembly

Question D We have displayed Register window and Disassembly window
D_1 (30 points) Can you determine the address of the next instruction that will be executed? If yes, please write it in hexadecimal notation.

Yes, we can determine the address of the next instruction as it is stated in the EIP of the register window as the EIP points to where the current instruction is.

012113BE

D_2 (30 points) What is the address of the element stored on TOP of the stack?

ESP always points towards the top of the stack therefore the address that ESP is pointing to is the top of the stack

00F3F730

D_3 (40 points) Can you determine the value of 32 bit word (in HEX) on top of the STACK (Intel processor)?

I looked at the disassembly window and saw 00F3F370 = 04112101 which I converted into binary
04112101 = 0000 0100 0001 0001 0010 0001 0000 0001

60/100

ZHANG		CHUE
Computer Science	or	***Computer Engineering***

```
Registers
EAX = CCCCCCCC EBX = 7F12F000 ECX = 00000000
EDX = 00000001 ESI = 01211104 EDI = 00F3F820
EIP = 012113BE ESP = 00F3F730 EBP = 00F3F820
EFL = 00000214
```

```
0x00F3F730  04 11 21 01  ..!.
0x00F3F734  04 11 21 01  ..!.
0x00F3F738  00 f0 12 7f  .ð..
0x00F3F73C  cc cc cc cc  ÌÌÌÌ
0x00F3F740  cc cc cc cc  ÌÌÌÌ
0x00F3F744  cc cc cc cc  ÌÌÌÌ
0x00F3F748  cc cc cc cc  ÌÌÌÌ
0x00F3F74C  cc cc cc cc  ÌÌÌÌ
0x00F3F750  cc cc cc cc  ÌÌÌÌ
0x00F3F754  cc cc cc cc  ÌÌÌÌ
0x00F3F758  cc cc cc cc  ÌÌÌÌ
0x00F3F75C  cc cc cc cc  ÌÌÌÌ
0x00F3F760  cc cc cc cc  ÌÌÌÌ
0x00F3F764  cc cc cc cc  ÌÌÌÌ
0x00F3F768  cc cc cc cc  ÌÌÌÌ
0x00F3F76C  cc cc cc cc  ÌÌÌÌ
0x00F3F770  cc cc cc cc  ÌÌÌÌ
0x00F3F774  cc cc cc cc  ÌÌÌÌ
0x00F3F778  cc cc cc cc  ÌÌÌÌ
0x00F3F77C  cc cc cc cc  ÌÌÌÌ
0x00F3F780  cc cc cc cc  ÌÌÌÌ
0x00F3F784  cc cc cc cc  ÌÌÌÌ
0x00F3F788  cc cc cc cc  ÌÌÌÌ
0x00F3F78C  cc cc cc cc  ÌÌÌÌ
0x00F3F790  cc cc cc cc  ÌÌÌÌ
0x00F3F794  cc cc cc cc  ÌÌÌÌ
0x00F3F798  cc cc cc cc  ÌÌÌÌ
0x00F3F79C  cc cc cc cc  ÌÌÌÌ
0x00F3F7A0  cc cc cc cc  ÌÌÌÌ
0x00F3F7A4  cc cc cc cc  ÌÌÌÌ
0x00F3F7A8  cc cc cc cc  ÌÌÌÌ
0x00F3F7AC  cc cc cc cc  ÌÌÌÌ
0x00F3F7B0  cc cc cc cc  ÌÌÌÌ
0x00F3F7B4  cc cc cc cc  ÌÌÌÌ
```

335/430 = 73