CSO I **Name:** _____

Fall 2023

Exam II: Attack of the Assembly

2023-11-8

Time Limit: 50 minutes Computing ID _____

Instructions:

1. This exam contains 10 pages (including this cover page) and 12 questions.
2. You have **50 minutes** to complete the examination.
3. Write your answers in this booklet. We scan this into GradeScope, so **please try to avoid writing on the backs of pages**. Please write the answers using a pen or write darkly.
4. If a question presents several options in a list, mark the bubble next to the one correct answer. All such questions on this test are single-select unless otherwise specified in a specific question.
5. You may not use a calculator or notes (should be obvious but needs to be said).
6. Because this assessment is being given in several places, we cannot fairly answer questions during it. If you find a question ambiguous or unclear, please explain that on the page by the question itself and we will consider your explanation during grading.
7. We will use the following data type sizes:

Types	size in bits
`char`	8
`short`	16
`int` and `float`	32
`long` and `double`	64

8. Function arguments are in (in order) `%rdi`, `%rsi`, `%rdx`, `%rcx`, `%r8`, `%r9`; return values are in `%rax`.
9. Please sign the below Honor Code statement.

I have neither given nor received aid on this exam.

Signature: _____

1 Memory, Structs and Pointer Fundamentals

1. Consider the following layout of memory assume that variables are stored in little endian in memory.

Registers	
General-Purpose	Value
EAX	0x0000000C
EBX	0x00000020
ECX	0x00200000
EDX	0x00000028
ESP	0x00000030
EBP	0x00B00000
ESI	0x00000001
EDI	0x000000FF

Main Memory Addresses			
Address	Code Segment	Address	Stack Segment
0x0010	0x00000000	0x0030	0x00000000
0x0014	0x00000004	0x0034	0x00000004
0x0018	0x00000008	0x0038	0x00000008
0x001C	0x0000000C	0x003C	0x0000000C
0x0020	0x00000010	0x0040	0x00000010
0x0024	0x00000014	0x0044	0x00000014
0x0028	0x00000018	0x0048	0x00000018
0x002C	0x00000034	0x004C	0x00000030

(a) (2 points) What is the result of executing `leaq (%eax, %ebx), %ecx`?

- ◯ `ecx = 0x0000034`
- √ `ecx = 0x0000002C`
- ◯ `ecx = 0x00000004`
- ◯ memory address `0x0000002C` becomes `0x00000004`
- ◯ memory address `0x0000002C` becomes `0x00200000`

(b) (2 points) Which of the following assembly instructions would result in a segmentation fault? **(Select all that apply) ***Assume that instruction from the previous question did not affect the state of memory. ***

- ◯ `leaq (%eax, %ebx, 8), %ecx`
- ◯ `movl (%esp), %ebx`
- ◯ `movl (%eax, %esp, 1), (%ecx)`
- √ `movl %eax(%ebx,%esi, 8), (%edx)`
- ◯ none of the above

Solution: Also will when accept this answer `movl (%eax, %esp, 1), (%ecx)`

(c) (2 points) Consider the following struct for the next two questions:

```
typedef struct {
    char info[8]
    int score;
}exam;
```

Which of the following are valid ways to declare a pointer to this struct?

- ◯ `struct exam *pointer;`
- ◯ `struct* exam pointer;`
- ◯ `typedef struct exam *pointer;`
- ◯ `typedef struct* exam pointer;`
- √ `exam *pointer;`
- ◯ `*exam pointer;`

(d) (2 points) If the pointer to struct is stored at memory location `0x04C`, what is the value of `pointer->info[4]` in hex? Assume that we are running on a 32-bit machine and **refer to memory layout at the beginning of this question.**

0x

Solution: 0x04

2 Flags

2. (2 points) What are the value for each flag given the following information (please write a value in the box) **SF**: sign flag, **ZF**: zero flag, **OF**: overflow flag, **CF**: carry flag

```
%rsi = 0x09
%rdi = 0x07
cmpq %rdi, %rsi
```

SF: **ZF:** **OF:** **CF:**

Solution:

SF: 0 **ZF:** 0 **OF:** 0 **CF:** 0

3. (2 points) What is the logical expression that determines if the `jne` branch is taken?

- ◯ `~(OF ^ SF)`
- √ `~ZF`

- ◯ `ZF & ~OF`
- ◯ `OF & ZF`
- ◯ none of the above.

3 Functions and Assembly

4. (2 points) Assume a function called

```
AnotherBanga(calm, down, x, y, p_1, p_2, z)
```

Which of the following are true when the function is called. **Select all that apply:**

- ◯ `calm` in `%rdx` and `p_2` in the `stack`
- ◯ `down` in `%rsi` and `y` in the `%r8`
- √ `x` **in** `%rdx` **and** `z` **in the** `stack`
- ◯ `y` in `%rsi` and `p_1` in the `%r8`
- ◯ `x` in `%rsi` and `p_2` in the `%r8`

5. (2 points) What line of C code does the assembly below implement? For example `return 4/x;` would be an example line

```
strange(int, int):
        movl    %edi, %eax
.L2:
        testl   %esi, %esi
        jle     .L5
        addl    %edi, %eax
        leal    -1(%esi), %esi
        jmp     .L2
.L5:
        ret
```

Select the function from the list below.

- ◯ `return y - y * x;`
- ◯ `return x - y * x;`
- ◯ `return y * x;`
- ◯ `return y * y;`
- ◯ `return x  - y * x;`
- √ `return x + y * x;`
- ◯ `return y / x;`
- ◯ `return x / y;`

4 Compilation

6. (2 points) What is the command on the terminal to compile a file called jungle.c with two levels of optimization that outputs an executable called a.out

   ◯ clang jungle.c

   ◯ clang jungle.c -o 2 a.out

   √ **clang -O2 jungle.c**

   ◯ jungle.c -o a.out 2

7. (3 points) Rearrange following lines so that they implement the visit function (i.e. `visit_Constant` or `visit_Add`) that generates the assembly for the following line

   ```
   x == 5
   ```

   Here is the implementation the visit method that is run.

   ```
   void visit_Compare(Node* node) {
       visit(self, node->left);
       visit(self, node->right);
       visit(self, node->ops);
   }
   ```

   Also assume that we have the following functions available `visit_constant` which pushes the constant onto the stack and `visit_Name` which looks up the value of variable and pushes it onto the stack. *****Clearly write the line numbers in the box below so that our AI can grade it*****

   ```
   1        printf("cmpq %%rdx %%rax");
   2        printf("popq %%rax");
   3        void visit_Eq(Node* node){
   4        }
   5        printf("popq %%rdx");
   ```


Solution:

3	5	2	1	4

8. (6 points) Assume the following Backus-Naur Form (BNF) grammar

```
<expression> ::= <term> | <variable> "=" <term>
<term> ::= <factor> | <factor> "+" <term> | <factor> "-" <term>
<factor> ::= <number> | <variable> | "(" <expression> ")"
<variable> ::= "x" | "y" | "z" | "w"
<number> ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

Fill in the parsing tree for the following statement:

```
x = x + 5
```

We have given you a word bank below (note: not every word will be used, and words can be used multiple times) *****Write clearly with a pen so that our AI can grade it*****

expression
+
=
-
variable
factor
identifier
number
x
5
term

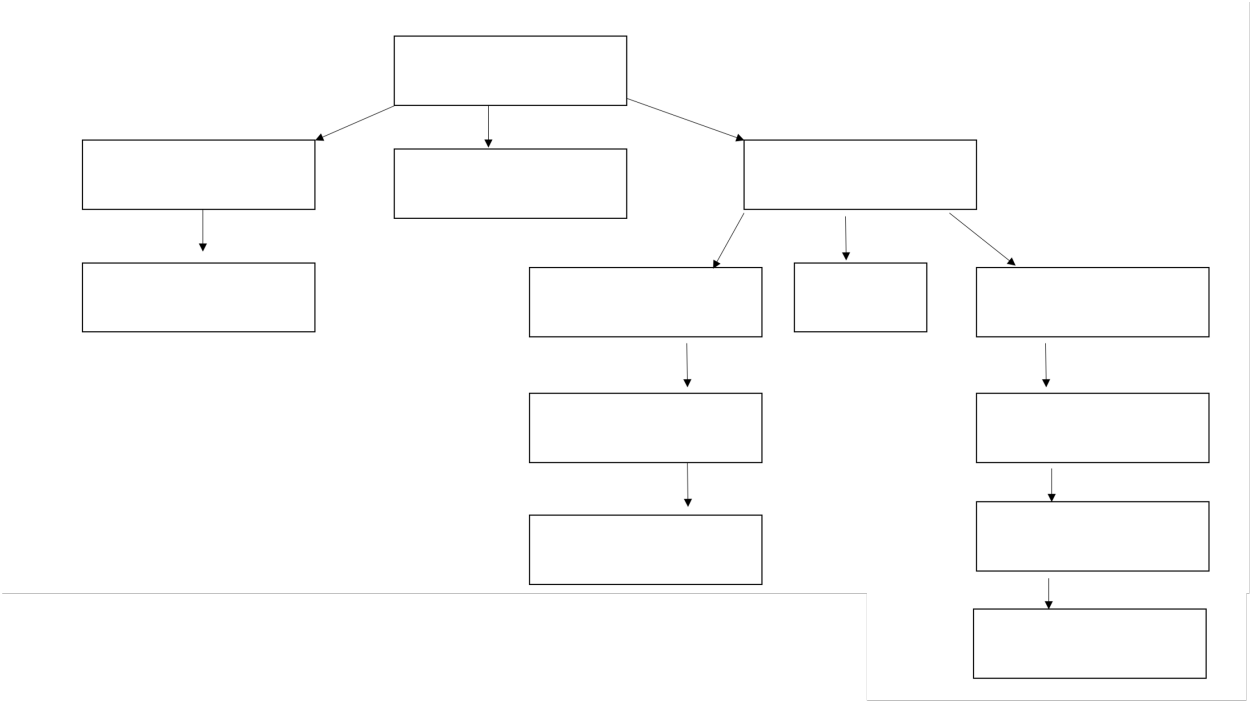

Figure 1: Parse tree

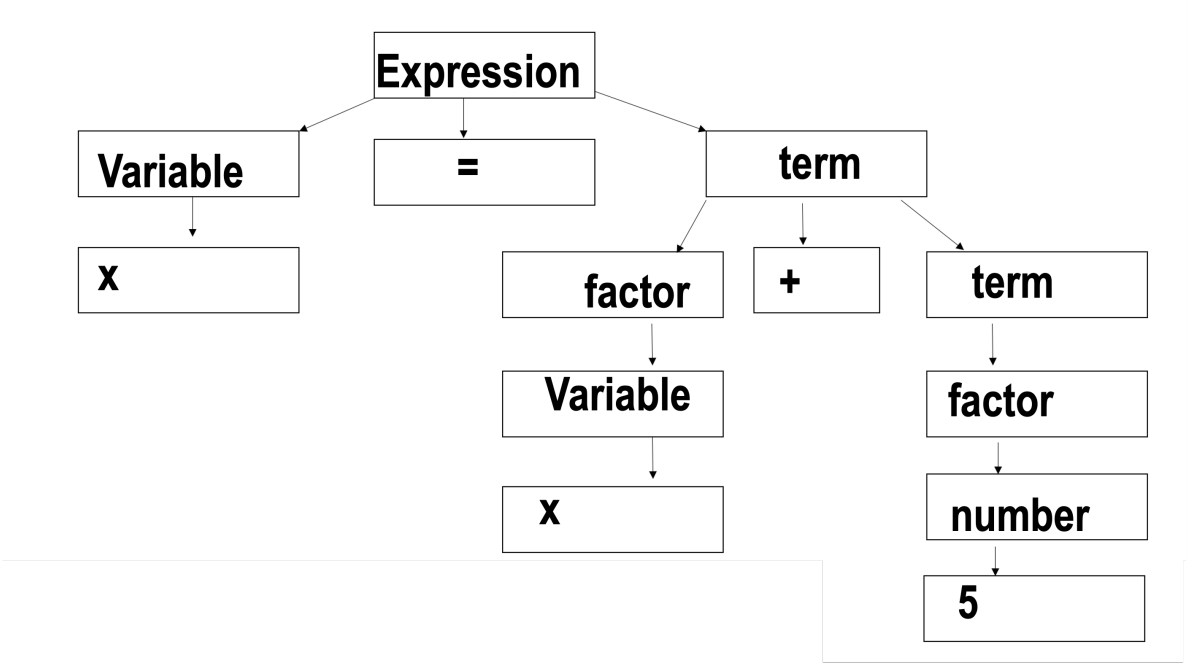

Figure 2: solution

5 Pointers

9. (2 points) What does the following program print out when it completes

```
int main() {
    int a = 10;
    int b = 20;
    int *ptrA = &a;
    int *ptrB = &b;
    *ptrA = *ptrA + *ptrB;
    *ptrB = *ptrA - *ptrB;
    *ptrA = *ptrA - *ptrB;
    printf(" a = %d, b = %d\n", a, b);
    return 0;
}
```

◯ a = 10 b = 20

√ **a = 20 b = 10**

◯ a = 30 b = 10

◯ a = 30 b = 20

◯ a = 20 b = 30

◯ a = 30 b = 30

◯ a = 10 b = 10

◯ a = 20 b = 20

◯ a = 0 b = 0

10. (5 points) What are the following values of i, j, *k, l, and *m after the following code is executed?

```
int i = 5;  //i is at memory address 0x2
int j = 7;  //j is at memory address 0xC
int *k = &i; //*k is at memory address 0xC0
int *l = &j; //*l is at memory address 0xC8
*k = 127;
*l = *k;
int *m = &i; //*m is at memory address 0xF0
*m = 3;
*l = *k * 5 + 9;
```

i: [ ] **j:** [ ] ***k:** [ ] **l:** [ ] ***m:** [ ]

Solution:

i: [3] **j:** [24] ***k:** [3] **l:** [0xC] ***m:** [3]

11. (4 points) Consider the following snippet

```
int intArray[] = {1, 2, 3, 4, 5};
char charArray[] = {'o','n','e','\0'};
void *arrayPointers[2] = {intArray, charArray};
```

Which of the following would correctly print the third character 'e', from the charArray? **(Select all that apply)**

- √ `printf("%c\n",*((char*)arrayPointers[1]+2));`
- ◯ `printf("%c\n",(char*)(*arrayPointers[1]+2));`
- ◯ `printf("%c\n", *(char)(*arrayPointers[1][2]));`
- ◯ `printf("%c\n", (char*)(arrayPointers[1][2]));`

12. (a) (1 point) How difficult was this exam?
    - ◯ too easy
    - ◯ easy
    - ◯ fair
    - ◯ difficult
    - ◯ too difficult

    (b) (1 point) An Extra Credit Opportunity for Everyone

    I'll be sharing my PowerPoint slides in a OneDrive folder. For every mistake you identify and correct, you'll earn up to `0.1%` added to your final grade, with a maximum limit of `1%`. Please note that if someone identifies and corrects an error before you, you won't be able to claim credit for it. We'd like to hear your thoughts on this option. Depending on your feedback, we will decide whether to keep it as an extra credit opportunity. Also, it's important to mention that awarding points will be at our discretion.
    - ◯ yes
    - ◯ no