hw7: Assembly Language 3

Due Apr 24 at 11:59pm **Points** 8 **Questions** 8
Available Apr 17 at 12am - Apr 25 at 11:59pm **Time Limit** 60 Minutes
Allowed Attempts 2

Instructions

Suggestions:

- Use your knowledge of what the compiler adds to functions to determine what parts of the code are most important. For example, questions that require code tracing don't necessarily require every assembly instruction to be traced.
- You may use the x86-cheat-sheet.pdf found in the Files section on course site for a reference of x86 assembly instructions.
- **Note:** The cheat sheet all or in part will be available for the final exam so spend time familiarizing yourself with the assembly instructions and their format.

This quiz was locked Apr 25 at 11:59pm.

Attempt History

	Attempt	Time	Score
KEPT	Attempt 2	59 minutes	7 out of 8
LATEST	Attempt 2	59 minutes	7 out of 8
	Attempt 1	36 minutes	4.33 out of 8

Score for this attempt: **7** out of 8
Submitted Apr 24 at 10:25pm
This attempt took 59 minutes.

Question 1

1 / 1 pts

```
#include <stdio.h>

typedef struct Point {
  int x;
  int y;
} Point;
```

```
int func(Point p1, Point p2);

int main(){
  Point p1, p2;
  p1.x = 8;
  p1.y = 10;

  p2.x = 2;
  p2.y = 6;

  printf("%d\n", func(p1, p2));

  return 0;
}
```

```
func:
    pushl %ebp
    movl  %esp, %ebp
    pushl %ebx
    subl  $24, %esp
    movl  20(%ebp), %eax
    movl  16(%ebp), %ecx
    movl  12(%ebp), %edx
    movl  8(%ebp), %ebx
    movl  %ebx, -16(%ebp)
    movl  %edx, -12(%ebp)
    movl  %ecx, -24(%ebp)
    movl  %eax, -20(%ebp)
    movl  -12(%ebp), %eax
    addl  -24(%ebp), %eax
    addl  $24, %esp
    popl  %ebx
    popl  %ebp
    ret
```

Given only the assembly code for `func`, the output of the program is:

Correct!

12

orrect Answers 12 (with margin: 0)

Question 2

1 / 1 pts

Consider the following program where the `zu` format specifier is used to display as an integer the result of `sizeof`.

```
#include <stdio.h>
#include <string.h>
```

```
typedef union {
    int *i1;
    int *i2;
}U;

int main() {
    U temp;
    int i = 50;
    temp.i1 = &i;
    printf("%zu and %d\n", sizeof(temp), *(temp.i2));
    return 0;
}
```

What is the output of the program above? Note <garbage value> in the choices below indicates output that isn't any of the literal values in the code above.

Correct!

● 4 and 50

○ 8 and <garbage value>

○ 4 and <garbage value>

○ Compilation error

○ 8 and 50

Question 3

1 / 1 pts

```
char myArray[8][4];
```

If, rather than row-major order, C stored data in column-major order the offset (in bytes) of `&myArray[5][2]` from `&myArray[0][0]` would be:

Correct!

21

orrect Answers

21 (with margin: 0)

Question 4

1 / 1 pts

Consider the following structure declarations:

```
struct st1 {
    char c[19];
    short s;
};
struct st2 {
    int i[3];
    char c[8];
    short s[1];
};
struct st3 {
    char c[5];
    int i[2];
    short s[3];
};
```

What is the total size of the structures st1, st2, and st3?

Correct!

● 22, 24, and 24

○ 22, 24, and 20

○ 20, 20, and 20

○ 24, 20, and 24

○ 22, 20, and 24

Question 5

1 / 1 pts

```
double *myArray[5][5][15];
```

If `&myArray[0][0][0]` is 0 then the value of `&myArray[3][4][5]` (in decimal) is:

Correct!

1,160

orrect Answers

1,160 (with margin: 0)

Question 6

0.67 / 1 pts

Assume that the starting address of doubles array A and integer index i are stored in registers %ecx and %edx, respectively. For each of the following expressions involving A, what would be its type and equivalent assembly code implementation, with the result being stored in %eax?

1. A+i+2 : double * and leal 16(%ecx, %edx, 8), %eax
2. *(A+i-2) : double and movl -16(%ecx, %edx, 8), %eax
3. &A[8]-A : double * and movl %ecx, %ebx; neg %ebx; leal 48(%ecx, %ebx), %eax

Answer 1:

Correct!

double * and leal 16(%ecx, %edx, 8), %eax

Answer 2:

Correct!

double and movl -16(%ecx, %edx, 8), %eax

Answer 3:

orrect Answer

int and movl $0x8, %eax

ou Answered

double * and movl %ecx, %ebx; neg %ebx; leal 48(%ecx, %ebx), %eax

Question 7

1 / 1 pts

Assume variable `a` is stored at memory address 0x80490000. What is the value stored in the pointer variable `ptr` for the following cases?

Case 1

```
char a[16];
char *ptr = &a[0];
```

Case 2

```
struct Point{
  int z;
  int y;
  int x;
};

struct Point a;
int *ptr = &a.x
```

Case 3

```
struct Node{
  int count;
  int nums[5];
};

struct Node a;
int *ptr = a.nums;
```

Correct!

Case 1

0x80490000

Correct!

Case 2

0x80490008

Correct!

Case 3

0x80490004

Other Incorrect Match Options:

- 0x80490018
- 0x80490020
- 0x80490014
- 0x80490010

Question 8

0.33 / 1 pts

Consider the following code fragment where the right-hand side of the assignments in `func` are to be completed by answering this question:

```
typedef struct point {
    float *p;
    struct {
        float x;
        float y;
    } s;
    struct point* next;
} point;

void func(point *sp1, point* sp2) {
    sp2->p = A;
    sp2->s.x = B;
    sp2->s.y = C;
    ...
}

int main() {
    point *init, *final;
    // assume init has been initialized to some value;
    func(init, final);
    return 0;
}
```

The compiler generates the following assembly code for the body of `func`:

```
pushl %ebp
movl %esp,%ebp
movl 0x8(%ebp),%eax
leal 0x8(%eax),%edx
movl 0xc(%ebp),%eax
movl %edx,(%eax)
movl 0x8(%ebp),%eax
movl 0x4(%eax),%eax
movl 0xc(%ebp),%edx
movl %eax,0x4(%edx)
movl 0x8(%ebp),%eax
movl 0x4(%eax),%eax
movl 0x8(%ebp),%ecx
addl 0x8(%ecx),%eax
movl 0xc(%ebp),%edx
movl %eax,0x8(%edx)
movl 0x8(%ebp),%eax
movl 0xc(%ebp),%edx
movl %edx,0xc(%eax)
movl 0xc(%ebp),%eax
movl $0x0,0xc(%eax)
popl %ebp
ret
```

Complete the first two assignment statements by answering the following:

The value of **A** is [Select]

The value of **B** is sp1->s.x

The value of **C** is [Select]

Answer 1:

orrect Answer &sp1->s.y

ou Answered sp1->p

Answer 2:

Correct! sp1->s.x

Answer 3:

orrect Answer sp1->s.x + sp1->s.y

ou Answered sp1->s.y

Quiz Score: **7** out of 8