CMSC 313 Fall2009
Midterm Exam 2
Section 01

Nov 11, 2009

Name__________________________________Score __________ out of 70

UMBC Username ____

Notes:

a. Please write clearly. Unreadable answers receive no credit.
b. For TRUE/FALSE questions, write the word TRUE or the word FALSE. T and F will result in a 2-point deduction.
c. There are no intentional syntax errors in any code provided with this exam. If you think you see an error that would affect your answer, please bring it to my attention.

TRUE/FALSE - 2 points each.
Write the word TRUE or the word FALSE in each blank

1. __________________ All switch statements are implemented with a jump table.

2. __________________ None of the IA32 instructions differentiate between sign and unsigned integer types.

3. __________________ When a function is executing, its stack frame is always the same size.

4. __________________ The last thing a function caller must do before executing its call instruction is to save its registers on the stack.

5. __________________ The first thing a function that is called must do is save the caller's frame pointer on the stack.

6. __________________ Given the declaration `short S[10][6]`, the memory address of `S[r][c]` can be calculated as `S + 12*(r-1) + 2*(c-1)`

7. __________________ All members of a union have the same byte offset relative to the start of the union.

8. __________________ The size of a function's stack frame can change while the function is executing.

9. __________________ The instruction pointer (%eip) contains the address of the next instruction to be executed.

10.__________________ The C compiler will check for an invalid array index if the size of the array is known at compiler time.

11. (10 points)

Consider the source code below, where M and N are constants declared with #define.

```
int array1[M][N];
int array2[N][M];

int copy(int i, int j)
{
     array1[i][j] = array2[j][i];
}
```

Suppose the above code generates the following assembly code:

```
copy:
     pushl %ebp
     movl %esp,%ebp
     pushl %ebx
     movl 8(%ebp),%ecx
     movl 12(%ebp),%ebx
     leal (%ecx,%ecx,8),%edx
     sall $2,%edx
     movl %ebx,%eax
     sall $4,%eax
     subl %ebx,%eax
     sall $2,%eax
     movl array2(%eax,%ecx,4),%eax
     movl %eax,array1(%edx,%ebx,4)
     popl %ebx
     movl %ebp,%esp
     popl %ebp
     ret
```

What are the values of M and N?

M	**N**

12. (6 points)

Consider the following code fragment containing the incomplete definition of a data type matrix_entry with 4 fields.

```
struct matrix_entry{

    ____________________ a;

    ____________________ b;

                     int c;

    ____________________ d;
};

struct matrix_entry matrix[2][5];

int return_entry_c (int i, int j){
    return matrix[i][j].c;
}
```

Complete the above definition of matrix_entry so that the following assembly code could be generated from it on a Linux/x86 machine:

```
return_entry_c:
    pushl %ebp
    movl %esp,%ebp
    movl 8(%ebp),%eax
    leal (%eax,%eax,4),%eax
    addl 12(%ebp),%eax
    sall $4,%eax
    movl matrix+4(%eax),%eax
    movl %ebp,%esp
    popl %ebp
    ret
```

Note that there are multiple correct answers. Choose your answers from the following types, assuming the following sizes and alignments

TYPE	SIZE	ALIGNMENT
char	1	1
short	2	2
int	4	4
double	8	4

13. (10 points)

Consider the following assembly representation of a function foo containing a for loop:

```
foo:
    pushl %ebp
    movl %esp,%ebp
    pushl %ebx
    movl 8(%ebp),%ebx
    leal 2(%ebx),%edx
    xorl %ecx,%ecx
    cmpl %ebx,%ecx
    jge .L4
.L6:
    leal 5(%ecx,%edx),%edx
    leal 3(%ecx),%eax
    imull %eax,%edx
    incl %ecx
    cmpl %ebx,%ecx
    jl .L6
.L4:
    movl %edx,%eax
    popl %ebx
    movl %ebp,%esp
    popl %ebp
    ret
```

Fill in the blanks to provide the functionality of the loop:

```
int foo(int a)

    int i;
    int result = ______________;

    for( _____________; _______________; i++ ) {

        ____________________;

        ____________________;
    }

    return result;
}
```

14. (6 points)

Match each of the assembler routines on the left with the equivalent C function on the right. Write the name of the function in the box in the table below

```
foo1:
     pushl %ebp
     movl %esp,%ebp
     movl 8(%ebp),%eax
     sall $4,%eax
     subl 8(%ebp),%eax
     movl %ebp,%esp
     popl %ebp
     ret
foo2:
     pushl %ebp
     movl %esp,%ebp
     movl 8(%ebp),%eax
     testl %eax,%eax
     jge .L4
     addl $15,%eax
     .L4:
     sarl $4,%eax
     movl %ebp,%esp
     popl %ebp
     ret
foo3:
     pushl %ebp
     movl %esp,%ebp
     movl 8(%ebp),%eax
     shrl $31,%eax
     movl %ebp,%esp
     popl %ebp
     ret
```

```
int choice1(int x)
{
     return (x < 0);
}

int choice2(int x)
{
     return (x << 31) & 1;
}

int choice3(int x)
{
     return 15 * x;
}

int choice4(int x)
{
     return (x + 15) /4
}

int choice5(int x)
{
     return x / 16;
}

int choice6(int x)
{
     return (x >> 31);
}
```

Assembly Code	**Function**
foo1	
foo2	
foo3	

15. (10 points):

This problem tests your understanding of the ***stack discipline and byte ordering.*** Here are some notes to help you work the problem:

- `strcpy(char *dst, char *src)` copies the string at address `src` (including the terminating '`\0`' character) to address `dst`. It does **not** check the size of the destination buffer.
- Recall that Linux/x86 machines are Little Endian.
- You will need to know the hex values of the following characters:

Character	**Hex Value**	**Character**	**Hex Value**
'a'	0x61	'f'	0x66
'b'	0x62	'g'	0x67
'c'	0x63	'h'	0x68
'd'	0x64	'i'	0x69
'e'	0x65	'\0'	0x00

```
/* copy string x to buf */
void foo(char *x) {
      int buf[1];
      strcpy((char *)buf, x);
}

void callfoo() {
foo("abcdefghi"); }
```

Here is the corresponding machine code on a Linux/x86 machine:

```
080484f4 <foo>:
080484f4: 55
      pushl %ebp
080484f5: 89 e5
      movl %esp,%ebp
080484f7: 83 ec 18
      subl $0x18,%esp
080484fa: 8b 45 08
      movl 0x8(%ebp),%eax
080484fd: 83 c4 f8
      addl $0xfffffff8,%esp
08048500: 50
      pushl %eax
08048501: 8d 45 fc
      leal
0xfffffffc(%ebp),%eax
08048504: 50
      pushl %eax
08048505: e8 ba fe ff ff
      call 80483c4 <strcpy>
0804850a: 89 ec
      movl %ebp,%esp
0804850c: 5d
      popl %ebp
0804850d: c3
      ret
08048510 <callfoo>:
08048510: 55
      pushl %ebp
08048511: 89 e5
      movl %esp,%ebp
08048513: 83 ec 08
      subl $0x8,%esp
08048516: 83 c4 f4         addl
$0xfffffff4,%esp
08048519: 68 9c 85 04 08
      pushl $0x804859c # push
string address
0804851e: e8 d1 ff ff ff
      call 80484f4 <foo>
08048523: 89 ec
      movl %ebp,%esp
08048525: 5d
      popl %ebp
08048526: c3
      ret
```

Now consider what happens on a Linux/x86 machine when `callfoo` calls `foo` with the input string "abcdefghi".

a. List the contents of the following memory locations immediately after strcpy returns to foo. Each answer should be an unsigned 4-byte integer expressed as 8 hex digits.

`buf[0] = 0x`__________________________

`buf[1] = 0x`__________________________

`buf[2] = 0x`__________________________

b. Immediately before the ret instruction at address 0x0804850d executes, what is the value of the frame pointer register %ebp?

%ebp = 0x____

c. Immediately after the ret instruction at address 0x0804850d executes, what is the value of the program counter register %eip?

%eip = 0x____

16. (4 points)

Consider the following C functions on the left and the assembly code on the right

```
int fun1(int a, int b)
{
     if (a < b)
          return a;
     else
          return b;
}

int fun2(int a, int b)
{
     if (b < a)
          return b;
     else
          return a;
}

int fun3(int a, int b)
{
     unsigned ua = (unsigned) a;
     if (ua < b)
          return b;
     else
          return ua;
}
```

```
  pushl %ebp
  movl %esp,%ebp
  movl 8(%ebp),%edx
  movl 12(%ebp),%eax
  cmpl %eax,%edx
  jge .L9
  movl %edx,%eax
.L9:
  movl %ebp,%esp
  popl %ebp
  ret
```

Which of the functions above on the left generated the assembly language above on the right? Write the name of the function in the box on the right.

17. (4 points)
Consider the following C functions and assembly code:

```
int fun4(int *ap, int *bp)
{
     int a = *ap;
     int b = *bp;

     return a+b;
}

int fun5(int *ap, int *bp)
{
     int b = *bp;
     *bp += *ap;

     return b;
}

int fun6(int *ap, int *bp)
{
     int a = *ap;
     *bp += *ap;

     return a;
}
```

```
pushl %ebp
movl %esp,%ebp
movl 8(%ebp),%edx
movl 12(%ebp),%eax
movl %ebp,%esp
movl (%edx),%edx
addl %edx,(%eax)
movl %edx,%eax
popl %ebp
ret
```

Which of the functions above on the left generated the assembly language above on the right? Write the name of the function in the box.