Full Name:______________________________________

CAS CS210 Computer Systems, Fall 2014

SOLUTIONS: MidTerm 1: Program Representation

Thursday Nov 6, 2014

Instructions:

- Make sure that your exam is not missing any sheets, then write your full name on the front.
- Write your answers in the space provided below the problem. If you make a mess, clearly indicate your final answer.
- Do your rough work in a work book provided. You do not need to hand in your work books.
- You may find it convenient to unstaple the exam. However please restaple in the correct order before handing it in.
- You may use your 1 page of notes that you brought with you.
- The exam has a maximum score of 43 points.
- You have 75 minutes to answer all questions. Good luck!

1 (3):
2 (3):
3 (3):
4 (10):
5 (8):
6 (10):
7 (6):
BONUS (4):
TOTAL (43):

Problem 1. (3 points):

Match each of the assembler routines on the left with the equivalent C function on the right.

```
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);
}
```

Fill in your answers here:

foo1 corresponds to choice ________. 1

`Answer:  Choice 3`

foo2 corresponds to choice ________. 1

`Answer:  Choice 5`

foo3 corresponds to choice ________. 1

`Answer:  Choice 1`

Problem 2. (3 points):

Consider the following C functions and assembly code:

```
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 compiled into the assembly code shown?

`Answer: fun1` **3**

Problem 3. (3 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 compiled into the assembly code shown?

`Answer: fun6` 3

Problem 4. (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;
}
```

```
int foo(int a)
{
    int i;         2
    int result = a + 2;
          2     2
    for (i=0; i < a; i++) {
result += (i + 5);  2
result *= (i + 3);      2
    }
    return result;
}
```

The next problem concerns the following C code:

```
/* 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
```

Problem 5. (8 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

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.

2

`buf[0] = 0x______________ Answer: buf[0] = 0x64636261`

2

`buf[1] = 0x______________ Answer: buf[1] = 0x68676665`

2

`buf[2] = 0x______________ Answer: buf[2] = 0x08040069`

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

2

`%ebp = 0x______________ Answer: ebp = 0x68676665`

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

2

`%eip = 0x______________ Answer: eip = 0x08040069`

Problem 6. (10 points):

Consider the following C declaration:

```
struct Node{
   char c;
   double value;
   struct Node* next;
   int flag;
   struct Node* left;
   struct Node* right;
};

typedef struct Node* pNode;

/* NodeTree is an array of N pointers to Node structs */
pNode NodeTree[N];
```

A. Using the template below (allowing a maximum of 32 bytes), indicate the allocation of data for a `Node` struct. Mark off and label the areas for each individual element (there are 6 of them). Cross hatch the parts that are allocated, but not used (to satisfy alignment).

Assume the alignment rules as follows.

Type	Size (bytes)	Alignment (bytes)
char	1	1
short	2	2
unsigned short	2	2
int	4	4
unsigned int	4	4
double	8	4

Clearly indicate the right hand boundary of the data structure with a vertical line.

```
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                                                                                                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+


Answer
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| c   X   X   X   d   d   d   d   d   d   d   d   p   p   p   p   i   i   i   i   p   p   p   p   p   p   p   p | XXXXXXXXXXXXXX
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
  c             |            value              |       next    |      flag     |      left     |      right    |

    1                    1                          1         1           1            1
```

B. For each of the four C references below, please indicate which assembly code section (labeled A – F) places the value of that C reference into register `%eax`. If no match is found, please write “NONE” next to the C reference.

The initial register-to-variable mapping for each assembly code section is:

```
 %eax = starting address of the NodeTree array
 %edx = i
```

```
-----------------------------------------------------------------
C References:
```

2 1. ______ NodeTree[i]->flag

2 2. ______ NodeTree[i]->next->next->flag

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

Linux/IA32 Assembly:

A.      sall $2, %edx                   B.  sall $2,%edx
        leal (%eax,%edx),%eax               leal (%eax,%edx),%eax
        movl 16(%eax),%eax                  movl (%eax),%eax
                                            movl 24(%eax),%eax
                                            movl 20(%eax),%eax
                                            movl 20(%eax),%eax

C:      sall $2,%edx                    D:  sall $2,%edx
        leal (%eax,%edx),%eax               leal (%eax,%edx),%eax
        movl 20(%eax),%eax                  movl (%eax),%eax
        movl 20(%eax),%eax                  movl 16(%eax),%eax
        movsbl (%eax),%eax

E:      sall $2, %edx                   F:  sall $2, %edx
        leal (%eax,%edx),%eax               leal (%eax,%edx),%eax
        movl (%eax),%eax                    movl (%eax),%eax
        movl 16(%eax),%eax                  movl 12(%eax),%eax
        movl 16(%eax),%eax                  movl 12(%eax),%eax
        movl 20(%eax),%eax                  movl 16(%eax),%eax

Answer: 1:D, 2:F
```

Problem 7 (6 Points):

Please answer each of the following questions. Be clear, concise and complete.

1. What is the value of an uninitialized local variable (e.g., int x;)?

2

Answer: Unknown as the stack memory for the containing call frame will have arbitary values in it.

2. How are strings represented in C?

2

Answer: As a NULL terminated array of bytes (chars). There is always one additional element that contains a 0 or NULL value that indicates the end of the array.

3. What is stack overflow (besides the website)?

2

Answer: When the memory outside of the current call frame is overwritten.

1 BONUS (Points 4):

Consider the following fragment of IA32 code from the C standard library:

```
0x400446e3 <malloc+7>: call   0x400446e8 <malloc+12>
0x400446e8 <malloc+12>: popl   %eax
```

After the popl instruction completes, what hex value does register %eax contain?

4

```
Answer: %eax = 0x400446e8
```