2011 Fall Final "Quiz"

Every 3rd Question

Question 3:

c)

$$\begin{array}{r} 5696C1B_{16} \\ +\ DA778_{16} \\ \hline 5771393_{16} \end{array}$$

Question 4:

a) a = 9
b) &a = 1016
c) b = 20
d) p = 1016
e) *p = 9
f) &p = 1008
g) q = 1012
h) *q = 20
i) cp = 1015
j) &cp = 1000

cp	1000	1015
q	1004	1012
p	1008	1016
b	1012	20
a	1016	9

Question 9:

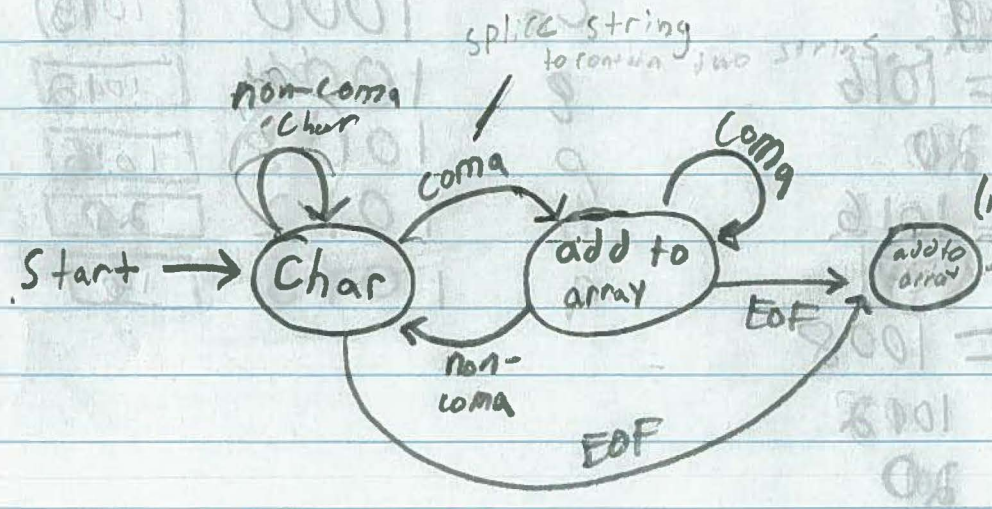
a) %ebp is $0x3E8_{16}$ (1000_{10})
b) %esp is $0x3C0_{16}$ (960_{10})
c) X's location is first
d) Y's location is second
e) T's Probable location is Third
f) location of Return's value is Last

Question 12:

Function: Char **split(char *s){

go through string characters; (String reading
upon arrival of a coma then a space put all previous
Char into array

once at end of file add all char. to array and null to array
and end Program.



CIS 2107
Computer Systems and Low-Level Programming
Fall 2011
Final

December 15, 2011

Name:

Page	Points	Score
1	10	
2	10	
3	9	
4	10	
6	5	
7	10	
8	10	
9	6	
10	10	
11	10	
12	10	
Total:	100	

Instructions

The exam is closed book, closed notes. You may *not* use a calculator, cell phone, etc.

For each of the questions of this quiz, you can assume the following sizes for C data types:

type	bytes
char	1
short	2
int	4
long	8
float	4
double	8
void*	4

(The answers are repeated at the top of this page for your convenience. They are the same as the answers on the previous page.)

A L1	H BSS	or program counter	U locality
B loader	I SRAM	O flash	V preprocessor
C nanoseconds	J compiler	P sector	W symbol table
D cylinder	K track	Q heap	X CPU
E L2	L EAX	R magnetic disk	Y EBP
F data	M seconds	S assembler	Z milliseconds
G DRAM	N instruction pointer	T stack	

(1 point) (k) Area of memory used for uninitialized global variables.

(k) _____

(1 point) (l) Tendency for programs to access multiple objects in a block.

(l) _____

(1 point) (m) Translates assembly language programs into machine language.

(m) _____

(1 point) (n) Area of memory used for initialized global variables.

(n) _____

(1 point) (o) In a collection of disk platters, a set of tracks equidistant from the center of the platter.

(o) _____

(1 point) (p) Processor register that contains the address of the next instruction to be executed.

(p) _____

(1 point) (q) Contains the return value of functions which return ints.

(q) _____

(1 point) (r) The larger but slower cache. Still much faster than main memory.

(r) _____

(1 point) (s) Time it takes to read from disk.

(s) _____

(1 point) (t) Time to read from on-CPU cache.

(t) _____

points: _____
out of a possible 10

exam continues ...

(2 points) 2. What is $111101011010_2 + 1011111011_2$ in base 2?

	1	1	1	1	0	1	0	1	1	0	1	0_2
+			1	0	1	1	1	1	1	0	1	1_2
	1	0	0	1	0	0	1	0	1	0	0	1

(2 points) 3. What is $5696C1B_{16} + DA778_{16}$ in base 16?

	5	6	9	6	C	1	B_{16}
+			D	A	7	7	8_{16}

(5 points) 4. The hex value 0x411a0000 is stored in a 32-bit C float variable. What floating point number does this value represent? It's perfectly OK if your answer includes fractions. (For example, if the correct answer were 2.75, you could also write "2 and $\frac{3}{4}$".)

points: _____
out of a possible 9

exam continues ...

(10 points) 5. **Some bit operations.** If we have `char x = 0x5C, y = 0xA9;`, what is the result of the following operations? Your answer must be in the form of exactly two hex digits[1].

(a) x|y

(a) 0xFD

(b) x||y

(b) 0x01

(c) x<<2

(c) 0x00

(d) ~x

(d) 0xA3

(e) ~~x

(e) 0x5C

(f) x&0x0F

(f) 0x0C

(g) x^y

(g) 0xF5

[1]Ignore the possibility of promotion to 32-bit ints. Behave as though we're living in the land of 8-bit arithmetic.

(h) x&&1

(h) 0x01

(i) -x

(i) 0xA3

(j) x-y

(j) 0x33

(k) !!x

(k) 0x01

(l) x<<1

(l) 0x38

(m) x&y

(m) 0x08

(n) x^y^y

(n) 0x5C

(o) x|0

(o) 0x5C

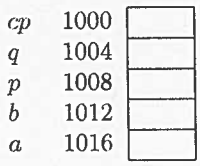
(5 points) 6. If I have the following:

```
int main(void)
{
  int a=10;
  int b=20;

  int *p=&a;
  int *q=p;
  char *cp = (char*)q;

  (*p)--;
  q--;
  cp--;
```

and memory is laid out like this:



what do you see if you print:

(a) a

(a) _____

(b) &a

(b) _____

(c) b

(c) _____

(d) p

(d) _____

(e) *p

(e) _____

(f) &p

(f) _____

(g) q

(g) _____

(h) *q

(h) _____

(i) cp

(i) _____

(j) &cp

(j) _____

points: _____
out of a possible 5

exam continues ...

(10 points) 7. Use the following code to answer the questions. Data sizes are specified on the cover of the exam.

```
1  struct Stuff {
2    int x;
3    int *p;
4    int A[10];
5  };
6
7  int main(void)
8  {
9    struct Stuff s;
10   int A[10];
11   int x, y;
12   char str[24];
13
14   x=10;
15   y=20;
16   A[0]=30;
17   strcpy(str, "almost quitting time");
18   s.x=40;
19   s.p=&y;
20   s.A[0]=50;
21
22   func01(A);
23   func02(str);
24   func03(str);
25   func04(s);
26
27   return 0;
28 }
29
30 void func01(int arr[]) {
31   arr[0]=3333;
32 }
33
34 void func02(char *s) {
35   strcpy(s, "yeah, winter break");
36 }
37
38 void func03(char *s) {
39   s = malloc(40);
40   strcpy(s, "how many more pages is this thing?");
41 }
42
43 void func04(struct Stuff s) {
44   s.x=4444;
45   *(s.p)=2222;
46   s.A[0]=5555;
47   s.p=malloc(sizeof(int));
48   *(s.p)=2020;
49 }
```

(a) How many bytes are passed to the function `func01( )`?

(a) _____

(b) How many bytes are passed to the function `func02( )`?

(b) _____

(c) How many bytes are passed to the function `func04( )`?

(c) _____

What is the value of each of the following after `func04( )` has been called?

(d) x

(d) _____

(e) y

(e) _____

(f) A[0]

(f) _____

(g) s.x

(g) _____

(h) s.A[0]

(h) _____

(i) *(s.p)

(i) _____

(j) str (What's the string?)

(j) _____

points: _____
out of a possible 10

exam continues ...

(10 points) 8. Write a C function which is passed an unsigned int x. The function returns 1 if there are an odd number of 1s in x's binary representation or 0 otherwise.

```
bool evens (unsigned int x){
    bool even = 0;
    while(x){
        even = !even;
        x = x & (x - 1);
    }
    return even;
}
```

points: _____
out of a possible 10

exam continues ...

9. Given the C function:

```
int func(int x, int y) {
  int t;

  ...

  return x+y-t;
}
```

Immediately before `func( )` is called, *i.e.*, immediately before the instruction `call func`, `%ebp` contains the value 1000_{10} and `%esp` contains the value 960_{10}. Before `func( )` exits (more precisely, just before the `leave` and `return` instuctions are executed), what is:

(1 point) (a) stored in `%ebp`?

(a) _____

(1 point) (b) stored in `%esp`?

(b) _____

(1 point) (c) the location of x?

(c) _____

(1 point) (d) the location of y?

(d) _____

(1 point) (e) the most likely location of `t`?

(e) _____

(1 point) (f) the location of the return value?

(f) _____

points: _____
out of a possible 6

exam continues ...

(10 points) 10. Write a C function equivalent to the following assembly (no credit for an answer containing inline assembly).

```
1           .section .text
2           .globl mystery
3           .type mystery, @function
4   mystery:
5           pushl %ebp
6           movl  %esp, %ebp
7           xorl  %eax, %eax
8           xorl  %ecx, %ecx
9           movl  8(%ebp), %edx
10  begin:
11          cmpl 12(%ebp), %ecx
12          jge  done
13          addl (%edx, %ecx, 4), %eax
14          incl %ecx
15          jmp  begin
16  done:
17          movl  %ebp, %esp
18          popl  %ebp
19          ret
```

points: _____
out of a possible 10

exam continues ...

(10 points) 11. Implement the function `void reverse(int A[ ], int len)`, which reverses the order of `A[ ]`, an array of `len` items. Do not use the `[ ]` operator. No credit will be given for solutions which use the `[ ]` operator, or which declare `len` or more elements of temporary storage.

```
void reverse(int A[ ], int len) {

Void reverse (int array[], int array_size) {
    int *pointer1 = array;
    int*pointer2 = array_size - 1;
    while(pointer1 < pointer2) {
        int temp = *pointer1;
        *pointer1 = *pointer2;
        *pointer2 = temp;
        pointer1++;
        pointer2--;
    }
}
```

(10 points) 12. A common way of storing a spreadsheet is comma-separated text. For example, the following line in a spreadsheet:

apple	banana	cherry	some fruit beginning with d

could be stored as "apple, banana, cherry, some fruit beginning with d". Write the function `char **split(char *s)` which is passed `s`, which is a string of comma-separated values, and returns an array of string containing the values in the line terminated by a NULL pointer. Using our current example, we'd return:

w[0]	apple
w[1]	banana
w[2]	cherry
w[3]	some fruit beginning with d
w[4]	NULL

`split( )` should return `NULL` on failure. Hint: if there are n commas in `s`, there will be $n + 1$ words. You may use any function in the Standard C Library.

points: _____
out of a possible 10

question 12 continues ...