United States Naval Academy
Electrical and Computer Engineering Department

EC312 - 6 Week Midterm (Version A) – Spring 2015

1. Do a page check: you should have 6 pages including this cover sheet.
2. You have 50 minutes to complete this exam.
3. An FE-approved calculator may be used for this exam. Calculators may not be shared.
4. This is a closed book and closed notes exam. You may use one single-sided hand-written page of notes.
5. Turn in your single-sided hand-written page of notes with your exam.
6. This exam may be given as a makeup exam to several midshipmen at a later time. No communication is permitted concerning this exam with anyone who has not yet taken the exam.

Name: _____SOLUTIONS____

Instructor: ____________________
**Question 1.** (4 pts) In the figure below, properly bias the NPN transistor by drawing in voltage sources (with polarity, i.e. + and -) and describing the relationship between $V_1$ and $V_2$.

\[ V_2 > V_1 \]

**Question 2.** (4 pts) Circle the correct answer in each pair of bold words:

An NPN transistor is properly biased when the Base-Emitter junction is forward-biased / reverse-biased, and the Base-Collector junction is forward-biased / reverse-biased.

**Question 3.** For the logic circuit shown below:

1) (3 pts) Annotate the diagram with the Boolean expression at the output of every gate.
2) (2 pts) Determine the Boolean expression for the output X.

\[ X = (A + C)B + BC \]

**Question 4.** (6 pts) The logic circuit depicted below has an unknown gate in the box annotated with a “?”. Use the Truth Table on the right to determine which one of our five fundamental gates is in the box.

**Unknown gate:** ___NOR___
**Question 5.** Answer the following questions pertaining to the memory diagram shown below.

(a) (3 pts) Fill in the missing portions of the addresses in the blanks on the left hand side.

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xBFFFF7FD</td>
<td>00</td>
</tr>
<tr>
<td>0xBFFFF7FE</td>
<td>00</td>
</tr>
<tr>
<td>0xBFFFF7FF</td>
<td>01</td>
</tr>
<tr>
<td>0xBFFFF800</td>
<td>16</td>
</tr>
<tr>
<td>0xBFFFF801</td>
<td>0A</td>
</tr>
<tr>
<td>0xBFFFF802</td>
<td>00</td>
</tr>
<tr>
<td>0xBFFFF803</td>
<td>00</td>
</tr>
<tr>
<td>0xBFFFF804</td>
<td></td>
</tr>
<tr>
<td>0xBFFFF805</td>
<td></td>
</tr>
<tr>
<td>0xBFFFF806</td>
<td></td>
</tr>
<tr>
<td>0xBFFFF807</td>
<td>84</td>
</tr>
<tr>
<td>0xBFFFF808</td>
<td></td>
</tr>
<tr>
<td>0xBFFFF809</td>
<td>04</td>
</tr>
<tr>
<td>0xBFFFF80A</td>
<td></td>
</tr>
</tbody>
</table>

(b) (4 pts) You are told that a variable of type `int` is stored at 0xBFFFF800. What is its decimal value? (Show all work.)

\[
0x00000A16 = 10 \times 16^2 + 1 \times 16^1 + 6 \times 16^0 = \mathbf{2582}
\]

(c) (4 pts) While running a program with the debugger gdb, you type the command `i r ebp` and get the following result:

```
(gdb) i r ebp
ebp 0xbffff80a
```

Suppose the next assembly language instruction which is successfully executed is

```
mov DWORD PTR [ebp-4], 0x08048425
```

Fill in the blank spaces in the memory diagram above on the right. Write in ‘G’ for any space that would still be an unknown garbage value.

**Question 6.** (7.5 pts) Circle TRUE or FALSE for each of the statements below.

(a) **TRUE/FALSE:** The register `eip` holds the address of the next instruction the CPU intends to execute.

(b) **TRUE/FALSE:** Machine language is high-level code that the compiler interprets.

(c) **TRUE/FALSE:** The stack pointer `esp` always points to addresses in the text segment.

**Question 7.** (3 pts) Suppose that a C program included the declaration statement

```
char team[5] = "Padres";
```

Which of the following is true? (Choose one)

(i) The program will definitely **not** compile (i.e. the compiler will throw errors).
(ii) The program will compile without errors, but `team` will initialize as “Padre”.
(iii) The program will compile without errors, but `team` will initialize as “Padre0”.
(iv) The program will compile without errors, but may exhibit unusual behavior since we have written beyond the range of the array `team`. 

**(iv)** The program will compile without errors, but may exhibit unusual behavior since we have written beyond the range of the array `team`. 

Question 8. (8 pts) What will be the output of this program when it is compiled and run?

```c
#include <stdio.h>
int main()
{
    int grades[6] = { 97, 92, 85 };  
    int j;
    for ( j = 0 ; j <= 6 ; j = j + 2 )
    {
        printf("Exam %d is %d \n", j+1, grades[j]);
    }
}
```

Exam 1 is 97
Exam 3 is 85
Exam 5 is 0
Exam 7 is GARBAGE VALUE

Question 9. Suppose a breakpoint is set at line 4 in the code below, and the program is run up until that point using gdb. The stack (with some blank spaces) is shown to the right. For the questions that follow, assume that the compiler doesn’t add any padding between items it stores on the stack.

```c
int square(int num)
{
    int result = num*num;
    return result;
}
```

```c
int main()
{
    int a = 2;
    int a_squared = 0;
    a_squared = square(a);
}
```

(a) (4 pts) If the variable `a` is stored at 0xBFFFF808 (as shown), fill in the remaining blank spaces in the memory diagram to the right.

(b) (4 pts) If you typed `i r ebp` at this point in program execution, what address will be given?

0xBFFFF7F8 (i.e. ebp for square’s stack frame)

(c) (4 pts) What address will eip point to after the function call is complete?

0x088304AC (i.e. the return address)

(d) (4 pts) Suppose you then set a breakpoint at line 12 and continue execution of the program up until that point. If you typed `i r ebp` at this point in program execution, what address will be given?

0xBFFFF80C (i.e. ebp for main’s stack frame)
**Question 10.** Consider the C program below, along with its corresponding output.

```c
#include<stdio.h>
int main()
{
    int *a_ptr, *b_ptr;
    int a = 10;
    b_ptr = &a;
    a_ptr = b_ptr;
    printf("The value of a is %d and its address is %x \n ", a , &a);
    printf("The pointer named a_ptr is at address %x \n\n" , &a_ptr);
}
```

The picture below shows a portion of main memory. Each box in the figure represents one byte of storage.

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xBFFFF86C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xBFFFF86D</td>
<td>0A</td>
<td>a</td>
</tr>
<tr>
<td>0xBFFFF86E</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>0xBFFFF86F</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>0xBFFFF870</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>0xBFFFF871</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xBFFFF872</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xBFFFF873</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xBFFFF874</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xBFFFF875</td>
<td>6D</td>
<td>a_ptr</td>
</tr>
<tr>
<td>0xBFFFF876</td>
<td>F8</td>
<td></td>
</tr>
<tr>
<td>0xBFFFF877</td>
<td>FF</td>
<td></td>
</tr>
<tr>
<td>0xBFFFF878</td>
<td>BF</td>
<td></td>
</tr>
<tr>
<td>0xBFFFF879</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(a) (4 pts) In the picture above, indicate in the Description column where the variable `a` is stored, and fill in the Value of `a` in the appropriate memory locations.

(b) (4 pts) In the picture above, indicate in the Description column where `a_ptr` is stored in memory.

(c) (4 pts) In the picture above, fill in the appropriate memory locations to indicate the Value of `a_ptr`.

**Question 11.** (3 pts) In which direction does the heap grow?

(i) From the bottom (larger memory address) up (to a smaller memory address).

(ii) From the top (smaller memory address) down (to a larger memory address).

(iii) It depends on the corresponding number and types of variables currently allocated on the stack.

(iv) The heap doesn’t grow; it has a fixed size.
**Question 12.** (3 pts) Fill in the blank in the code below to allocate space for 6 int variables on the heap.

```c
int *ptr1;
ptr1 = (int *) malloc(_24_);
```

**Question 13.** Consider the file listing shown below.

```
-rw-r-x--- 1 atwood instructor 6482 2014-01-29 02:54 gethappy.exe
```

(a) (2.5 pts) Who is the owner of this file?

`atwood`

(b) (2.5 pts) What permissions does the owner have?

Read, write, and execute

(c) (2.5 pts) What permissions do members of the group have?

Read and execute

**Question 14.** Consider the program below, named `AutoFry.c`.

```c
#include<stdio.h>

int main()
{
    int days_restriction = 60;
    int fry_level = 6000;
    char offender_name[20];
    int tours = 30;

    printf( "Enter offender’s last name: ");
    scanf( "%s", offender_name );
    printf("\n MIDN %s is awarded %d days of restriction and %d tours \n%n",offender_name, days_restriction, tours);
}
```

Assume that no padding (extra space) is created when variables are loaded on the stack.

(a) (6 pts) If an enterprising midshipman wants to use a buffer overflow attack to completely overwrite the value of the variable `days_restriction`, what is the minimum number of characters they would have to enter when prompted? **Justify your reasoning by including a diagram of the program stack.**

27 characters (since we need a total of 20+4+4 bytes, and a NULL will be automatically added to the characters that are typed)

(b) (4 pts) Is it possible to change the value of the variable named `tours` by performing a buffer overflow attack (i.e. by entering characters when prompted)? Why or why not? **Justify your reasoning.**

No. Since `tours` is declared after `offender_name`, it is stored “higher” on the stack, and therefore a buffer overflow from `offender_name` can never affect `tours`.

Turn in your equation sheet with your exam!