

Name: _____

Midterm Syllabus:

1. Number systems
 - a. Signed and unsigned representations
 - b. Conversion between different representations
 - c. Overflow condition
2. C Pointers
3. Dynamic Memory Allocation
4. C Data structures: Arrays, structs, multi-dimensional and multi-level arrays
5. Data in memory
 - a. Endian-ness
 - b. Heap or Stack?
 - c. Data alignment
6. ARM ISA basics
7. ARM arithmetic and bitwise instructions

Midterm Sample Questions

Section 1: Short Answers

- a) What evaluates to FALSE in C? What evaluates to TRUE?
- b) How do dynamically allocated data objects and local variables differ in terms of their lifetime?
- c) What is a dangling pointer?
- d) Give one example for i) type promotion and ii) type truncation in C.
- e) How is a multi-level array different from a multi-dimensional array?
- f) What do RISC and CISC stand for? Explain at least one difference between them.

Name: _____

2) Section 2: Number Representation

“There are 256 ways to personalize a Wendy’s burger”

(5 points) How many yes/no options (e.g., mustard/no mustard, lettuce/no lettuce, etc.) do you have to “personalize” your Wendy’s hamburger?

(5 points) If Wendy’s added two additional yes/no options, how many total ways would one now have to “personalize” your Wendy’s burger? Assume that these two options are totally independent of the 256 current ways of personalization.

(5 points) Assume that In-N-Out has N ways to personalize their burger. How many bits do they require to represent all of these personalization options?

(5 points) Knowing the secret In-N-Out menu expands the number of options by 17 times, i.e., there are $17N$ ways to personalize your burger if you know the secret menu options. How many bits does this require? How many qubits does this require? (Remember a qubit can have one of three states)

(15 points) Write the Condition bits for the following calculations (8-bit two’s complement numbers)

01101010
10110110
+ -----
Negative:
Zero:
Overflow:
Carry:

00111100
01010110
- -----
Negative:
Zero:
Overflow:
Carry:

10001011
01110110
+ -----
Negative:
Zero:
Overflow:
Carry:

Name: _____

(10 points) Deduce logically the condition bits (NZVC) for the following arithmetic operation: $54_{10} - 278_{10}$ when a 64-bit 2's complement format is used. You should not try to do the actual bitwise subtraction.

Section 3: Pointers and arrays

Q1: For parts a) and b) assume the following:

- I. The byte ordering is Little Endian
- II. The address of the first element of the array 'arr' is 0x00
- III. short is 2 bytes.

a) (10 points) Using a diagram show the byte level arrangement of the elements of the array 'arr' in memory. Use a hex format to represent the values stored in each byte of memory. What is the value of $*(*(arr+1)+1)$?

```
short arr[2][3]={ {0x01,0x02,0x03}, {0x04,0x05,0x06} };
```

b) (5 points) Consider the following C code

Name: _____

```
void print_elements (int i)
{
    int j;
    short arr[3]={0x01,0x02,0x03};
    char *ptr = (char *) arr;
    for (j=0; j<=i; j++)
        printf ("\n %x,",*(ptr+j));
}
```

What is the output when print_elements is called with an argument of 4 i.e.: print_elements(4);

Q2: More fun with pointers

For the C code below make the following assumptions

- i. The base address of array is 0x10000000
- ii. The byte ordering is Little Endian
- iii. int is 4 bytes

```
int array[5]={-1,1,2,3,4,5};
int main(){
    char *ptr1 = (((char *) array) + 5);
    char val;
    int *ptr2 = array + 3;
    val = *ptr1;
    return 0;
}
```

- i) **(10 points)** Show the byte level representation of array in memory. You have to mark the increasing direction of memory addresses.
- ii) **(5 points)** What is the value in *ptr2 at the end of the program?
- iii) **(5 points)** What is the value of the variable val at the end of the program?

Q3: (10 points) Draw a diagram to show the final relationship between pointers and pointees and the values in the array 'arr' when the following code is executed

```
main() {
```

Name: _____

```
char *p, *q;
char arr[] = "Doing well! Keep it up";

p = arr;
q = arr + 4;
p = q;
*q = *(p + 1);

}
```

Q4: C Strings: Consider the following definition of the function 'append'.

```
/*
Return the result of appending the characters in the string s2 to the string s1.
Assume enough space has been allocated for s1 to store
the extra characters.
*/
char* append (char s1[ ], char s2[ ])
{
    int s1len = strlen (s1);
    int s2len = strlen (s2);
    int k;
    for (k=0; k<s2len; k++) {
        s1[k+s1len+1] = s2[k];
    }
    return s1;
}
```

There are two bugs in the above code. Identify them. Explain why they are bugs and modify the code to correct them.

Q6: Dynamic Memory allocation: Consider the following declaration for the multi-level array, names:

```
char name_1[] = "John";
char name_2[] = "Paul";
char name_3[] = "Stephen";
char *names[3] = {name_1, name_2, name_3};
```

Create an equivalent multi-level array: dynamic_names that has all its elements and the data pointed to by its elements on the heap.

Section 4: ARM Arithmetic and Bitwise

Name: _____

Q1: (5 points) Perform the following multiply operation by writing one or more instructions in ARM. Assume the required inputs are available in registers. Use the following variable to register mapping in all cases: x : r0, y : r1. Multiply x by 25_{10} and store the result in y (Don't use the MUL/MLA instructions). How many bytes does your code need?

Q2: (15 points, 10 minutes) Copy the most significant byte of the variable y into the least significant byte of the variable x . All other bits in x should remain unchanged. For example, if x has the initial value 0x83123456 and y has the value 0xFE128A20, x should change to 0x831234FE. Assume the variable x is mapped to register r0, y is mapped to register r1. You can use other registers to store intermediate results. Comment each line. How many bytes does your code need?