

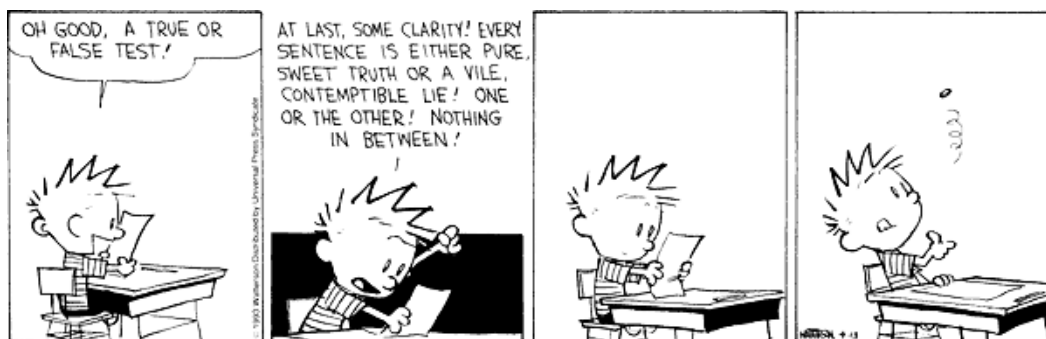
Name: _____

University of California, San Diego
CSE 30–Computer Organization and Systems Programming
Winter 2014 – Midterm
Dr. Diba Mirza

Name	
Student ID	

Name of person to your left	
Name of person to your right	

Please put your name on the top of every page and complete the all of the above information.



Problem	Possible	Score
1	10	
2	10	
3	10	
4	10	
5	20	
6	5 (extra credit)	
7	10	
8	10+5 (extra credit)	
Total	80	

Name: _____

Name: _____

Section I: (10 points) Short answers

1) Answer the following:

a) **(2 Point)** How do modern computers represent signed integer numbers?

2's Compliment

b) **(2 Points)** In C, an out of bound array access can cause an error during
runtime

i) Compile time ii) Run time iii) Both run time and compile time

c) **(2 Points)** What does the acronym ISA stand for? What does the ISA specify?

Instruction Set Architecture: Specifies the hardware/software interface. This includes the set of primitive (machine) instructions that can be executed on a processor and the registers available for programming in Assembly.

d) **(2 Points)** How many bytes are used to encode a single instruction in ARM (not the THUMB variant)?

4 bytes

e) **(2 Points)** What is the size of an ARM register?

4 bytes

Name: _____

Section II: Number Representation: 110000100001010100010101001111000001001000100

Problem 2 (10 points): Consider the binary pattern P: 0b10000101

a) **(6 points)** What is the decimal number represented by P for each of the following encodings:

- i) Unsigned: $2^7 + 2^2 + 2^0 = 133$
- ii) Sign and Magnitude: -5
- iii) 2's complement: $-2^7 + 5 = -123$

b) **(4 points)** Fill in the space provided below to represent 0b10000101 (P) in 2 bytes using a 2's complement encoding and hex notation. Each box holds 1 byte of information.

0xFF	0x85
MSB	LSB

Problem 3 (10 points): Bits and things

a) **(4 points)** How many colors can you represent with 2 qubits? $3^2 = 9$

Recall each qubit can represent 3 states

b) **(4 points)** How many bits (not qubits) would you need to represent four more colors?
 $9 + 4 = 13$ colors, Need at least 4 bits to represent 13 colors

c) **(2 points):** The color of a pixel on a computer screen is represented in memory by values for its red, green and blue components. If 4 bits are available to encode the red component, 3 bits are available to encode the green component and 1 bit is available to encode the blue component, how many colors can be represented for each pixel?
 $2^8 = 256$

Name: _____

Section III: Pointers

Problem 4: (10 points) 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.

```
short arr[2][3]={ {0xA,-2,0xB},{0xC,0xD,0xE}};
```

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) (4 points) Byte level representation of 'arr' in memory

0x0A	0x00	0xFE	0xFF	0x0B	0x00	0x0C	0x00	0x0D	0x00	0x0E	0x00
------	------	------	------	------	------	------	------	------	------	------	------

Increasing direction of memory addresses



Important steps to come up with above representation

- 1. Write 2 byte representation of each element in hex: arr[0] = 0x000A
- 2. Since Byte ordering is little endian Least Significant Byte would be in the lowest memory location
- 3. Represent -2 in a 2 byte 2's complement notation: 0b111111111111110 = 0xFFFE

b) (6 points) What is printed by each of these statements?

- I. `printf ("\\n %x ",*((arr+1)+2));` 0xE
- II. `printf ("\\n %x ",**arr);` 0xA
- III. `printf ("\\n %x ",*((unsigned char *) arr +2 ) );` 0xFE

Name: _____

Problem 5: (20 points) In HW 2, you were given a structure similar to the following to store a sorted list of integers:

```
#define MAXELEMENTS 4

struct List {
    short size;
    int sortedList[MAXELEMENTS]
};

typedef struct List list;
```

a) (8 points) Use a diagram to represent an instance of `list` in memory for the case when:

i. Data is aligned



2 bytes of zero padding after 'size' to make sure that the first element of the array starts at a memory location divisible by 4 (the size of int on a 32 bit machine)

- ii. **Data is not aligned**



***Note: You should mark the starting address of each element of the structure as well as the size of the element. Make the following assumptions:**

1. The architecture is 32 bit (short is 2 bytes and int is 4 bytes)
2. The starting address of the structure is 0x1FC

Name: _____

- b) (12 points) Recall that in the homework you were asked to implement a function `createList()`. This function creates an empty list and returns the starting address of the list structure. Consider the following implementation of `createList()`:

```
list * createList()
{
    list newList;
    newList.size =0;
    return (&newList);
}
```

- i) (2 points) Where is `newList` created in program memory?

On the stack

- ii) (10 points) Is the above implementation correct? If not, state why and provide a correct implementation in the space provided:

No, since the instance of the list 'newList' is created on the stack it is no longer available after the function returns which was not the intended implementation of `createList`

```
list * createList()
{
    list *newList = (list *) malloc (sizeof(list));
    if (newList !=NULL)
        newList->size =0;
    return(newList);
}
```

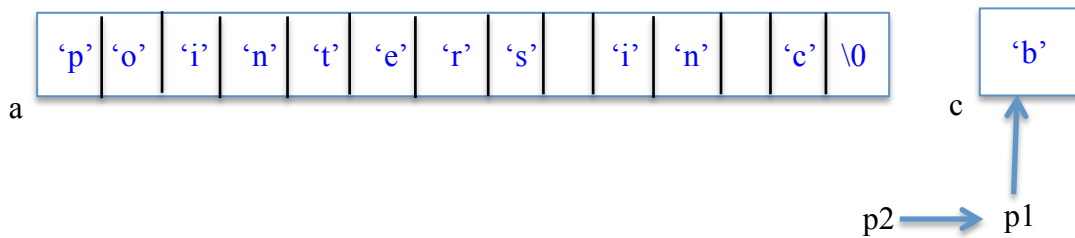
Name: _____

Problem 6 (Extra credit) (5 points) C Syntax and Semantics

Given the following declarations:

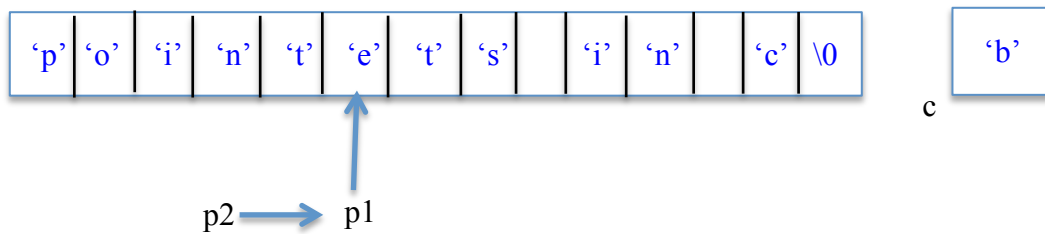
```
char a[14] = "pointers in c";  
char c = 'b';  
char *p1 = &c, **p2 = &p1;
```

- a) **(2 points)** Use a pointer diagram to show the relationship between the variables: 'a', 'c', 'p1', 'p2' after the above statements are executed.



- b) **(3 points)** Re-draw the pointer diagram and the contents of the array 'a' after the following statements are executed

```
p1 = a + 5;  
*(*p2 + 1) = 't';
```



Name: _____

Section IV: ARM Assembly

Problem 7: (10 points) Perform the following operations 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.

- a) **(5 points)** Negate x and store the result in y

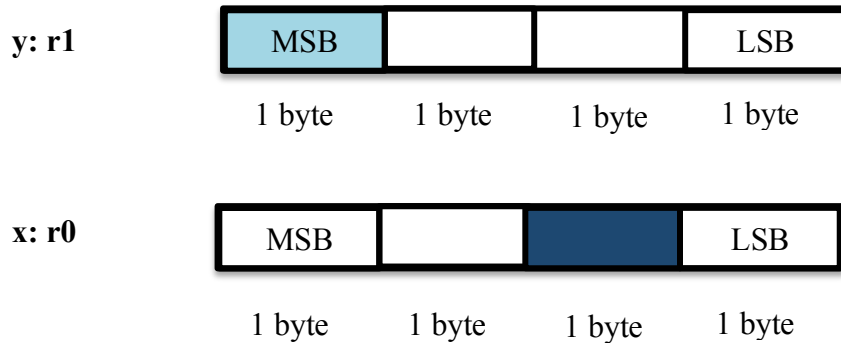
RSB r1, r0, #0; $r1 = 0 - r0$

- b) **(5 points)** Multiply x by 15_{10} and store the result in y (Don't use the MUL/MLA instructions)

RSB r1, r0, r0, LSL #4; $r1 = 16 * r0 - r0$

Name: _____

Problem 8: (10 points) Write ARM instructions to copy the most significant byte of the variable y (shown shaded in the picture) into the second least significant byte of the variable x (shown shaded in the picture) using bit-wise logical operations and shifts. All other bits in x should be unchanged. No control flow (e.g., loops/ifs) is needed. x is mapped to register $r0$, y is mapped to register $r1$. Extra credit (5 points) for using the fewest instructions.



Suggested Approach: Write down the steps that need to be done in plain English

1. Clear the 2nd least significant byte of $r0$, store back in $r0$
2. Zero out all bytes of $r1$ excluding the Most Significant byte that should be copied into $r0$
3. Align the MSB of $r1$ to the 2nd LSB of $r0$, store back in $r1$
4. OR $r0$ and $r1$ and store back in $r0$

```
MOV r3, 0xFF
BIC r0, r0, r3, LSL #8 ; step 1
```

```
LSR r1, r1, #24 ;step 2
```

```
LSL r1, r1, #8 ;step 3
ORR r0, r0, r1 ;step 4
```

Rewrite code to use min. no. of instructions (step 3 and 4 can be combined)

```
MOV r3, 0xFF
BIC r0, r0, r3, LSL #8 ; step 1
```

```
LSR r1, r1, #24 ;step 2
```

```
ORR r0, r0, r1, LSL #8 ; Combining step 3 and 4
```

Name: _____

Category	Example Instruction	Meaning
Arithmetic	ADD r0, r1, r2 SUB r0, r1, r2 RSB r0, r1, r2 ADD r0, r1, #100 MUL r0, r1, r2 MLA r0, r1, r2, r3	r0 = r1 + r2 r0 = r1 - r2 r0 = r2 - r1 r0 = r1 + 100 r0 = r1 * r2 r0 = (r1 * r2) + r3
Register Setting	MOV r0, r1 MOV r0, #100 MOV r1, r2, LSL #4 MOV r0, r1, LSR #8 MOV r0, r1, ROR #12 MVN r0, #0 MOV r0, r1, ASR #5	r0 = r1 r0 = 100 r1 = r2<<4 r0 = r1 >> 8 r0 = r1 rotated right 12 bits, r0=r1 rotated left 20 bits r0 = -1 r0 = r1/32
Data Transfer	LDR r0, [r1] STR r0, [r1] STR r1, [r0, r2, LSL #3] LDR r1, [r0, r2, LSL#3] STRB r0, [r1] LDRB r0, [r1] (LDRH, LDRB..etc for half word or byte)	r0 = Mem[r1] Mem[r1] = r0 Store data in r1 to <r0+r2*8> Load data in <r0+r2*8> to r1 Store byte Mem[r1]=r0 Load byte r0 = Mem[r1]
Branch	B Label CMP r0, r1 BEQ Label BNE Label BGE Label BGT Label BLE Label BLT Label	PC = Label Compare r0 and r1 and do... if (r0 == r1) go to Label if (r0 ≠ r1) go to Label if (r0 ≥ r1) go to Label if (r0 > r1) go to Label if (r0 ≤ r1) go to Label if (r0 < r1) go to Label
Logical Instructions	AND r0, r1, r2 ORR r0, r1, r2 EOR r0, r1, r2 BIC r0, r1, r2 RSB r0, r1, r2	r0 = r1 & r2 r0 = r1 r2 r0 = r1 ^ r2 r0 = r1 & (!r2) r0 = r1 - r2
Shifts and Rotates	LSL r0, r1, r3 LSR r0, r1, #5 ASR r0, r1, #5 ROR r0, r1, #12	r0 = r1<<r3 r0 = r1/32 (unsigned) r0 = r1/32 (signed) r0 = r1 rotated right 12 bits, r0=r1 rotated left 20 bits

Name: _____

--	--	--

Not all instructions are listed; any valid ARM instruction can be used.