Review Problem 4

◆ What would the results of this C++ code be in memory? Assume we start using memory at 1000.

```cpp
struct {char *a, *b;} foo;
foo *obj;
obj = new foo;
obj->a = new char[8];
obj->b = new char[4];
obj->a[1] = 'x';
obj->b[2] = 'y';
```
Array Example

/* Swap the kth and (k+1)th element of an array */
swap(int v[], int k) {
    int temp = v[k];
    v[k] = v[k+1];
    v[k+1] = temp;
}

// Assume v in X0, k in X1

Swap:

  LSL X2, X1, #3  // X2 = 8 * k
  ADD X2, X0, X2  // X2 = &v[k]
  LDUR X3, [X2, #10]  // get v[k]
  LDUR X4, [X2, #8]  // get v[k+1]
  STUR X4, [X2, #0]
  STUR X3, [X2, #8]

Memory:

<table>
<thead>
<tr>
<th>GPRs</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>X0: 928</td>
<td></td>
</tr>
<tr>
<td>X1: 10</td>
<td></td>
</tr>
<tr>
<td>X2:</td>
<td></td>
</tr>
<tr>
<td>X3:</td>
<td></td>
</tr>
<tr>
<td>X4:</td>
<td></td>
</tr>
</tbody>
</table>

Load

Store
Execution Cycle Example

PC: Program Counter
IR: Instruction Register

Note:
Word addresses
Instructions are 32b

General Purpose Registers

| X0:  | 928 |
| X1:  | 10  |
| X2:  | 801008 |
| X3:  |     |
| X4:  |     |

Memory

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>D3600C22</td>
</tr>
<tr>
<td>0004</td>
<td>8B020002</td>
</tr>
<tr>
<td>0008</td>
<td>8F400043</td>
</tr>
<tr>
<td>0012</td>
<td>F8408044</td>
</tr>
<tr>
<td>0016</td>
<td>F8400044</td>
</tr>
<tr>
<td>0020</td>
<td>F8408043</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>0A12170D34BC2DE1</td>
</tr>
<tr>
<td>1008</td>
<td>1111111111111111</td>
</tr>
<tr>
<td>1016</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>1024</td>
<td>0F0F0F0F0F0F0F0F</td>
</tr>
<tr>
<td>1032</td>
<td>FFFFFFFFFFFFFFFF</td>
</tr>
<tr>
<td>1040</td>
<td>FFFFFFFFFFFFFFFF</td>
</tr>
</tbody>
</table>

Instruction Fetch
Instruction Decode
Operand Fetch
Execute
Result Store
Next Instruction
Flags/Condition Codes

Flag register holds information about result of recent math operation
  Negative: was result a negative number?
  Zero: was result 0?
  Overflow: was result magnitude too big to fit into 64-bit register?
  Carry: was the carry-out true?

Operations that set the flag register contents:
  ADDS, ADDIS, ANDS, ANDIS, SUBS, SUBIS, some floating point.

Most commonly used are subtracts, so we have a synonym: CMP
  CMP X0, X1  same as SUBS X31, X0, X1
  CMPI X0, #15 same as SUBIS X31, X0, #15
Control Flow

Unconditional Branch – GOTO different next instruction

B START // go to instruction labeled with "START" label
BR X30 // go to address in X30: PC = value of X30

Conditional Branches – GOTO different next instruction if condition is true

1 register: CBZ (==0), CBNZ (!= 0)

CBZ X0, FOO // if X0 == 0 GOTO FOO: PC = Address of instr w/FOO label

2 register: B.LT (<), B.LE(<=), B.GE (>=), B.GT(>), B.EQ(==), B.NE(!=)

first compare (CMP X0, X1, CMPI X0, #12), then b.cond instruction

CMP X0, X1 // compare X0 with X1 - same as SUBS X31, X0, X1
B.EQ FOO // if X0 == X1 GOTO FOO: PC = Address of instr w/FOO label

// X0 = a, X1 = b, X2 = c

if (a == b)
    a = a + 3;
else
    b = b + 7;
c = a + b;

// set flags

CMP X0, X1 // branch if a!=b
B.NE ELSEIF
ADDI X0, X0, #3 // a = a + 3
B DONE // avoid else
ELSEIF:
ADDI X1, X1, #7 // b = b + 7
DONE: ADD, X2, X0, X1 // c = a + b