Please put out answers just to me in chat.

Mul I xe, x1, #1 - 1 → covext

Sub x0, x3, x1, li x3 = 0 → connected & effective

Sub x0, 0, x1

Sr x0, x9, x1

In assembly, set x0 to -x1.

Review Problem 2
Example: Take bits 6-4 of X0 and make them bits 2-0 of X1, zeros otherwise:
```
ANDI X0, X1, #7
LSL X0, X1, #4
LSE IF X0, XI, #1
```

Immediate Logical: AND, OR, EOR
Immediate (one input a constant)
Mathematic: ADD, SUB, MUL, DIV

(Note: Just subset of all instructions)

Basic Operations

```
A = B + C
A = B * C
A = B >> C
```

C/Java: A = B + C

C/Java: A = B * C

C/Java: A = B >> C
"Byte addressing" means that the index points to a byte of memory.

A memory address is an index into the array.

Viewed as a large, single-dimension array, with an address:

Memory Organization
Double-words and words are aligned.

2^61 double-words with byte addresses 0, 8, 16, ... 2^64-8
2^64 bytes with byte addresses from 0 to 2^64-1

Registers hold 64 bits of data.

<table>
<thead>
<tr>
<th>64 bits of data</th>
<th>64 bits of data</th>
<th>64 bits of data</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Word = 32 bits = 4 bytes
Double-word = 64 bits = 8 bytes
Bytes are nice, but most data items use larger units.

ARP 64 bit.
ARM: can do either – this class assumes Little-Endian.

Intel x86, DEC VAX, Altera Nios II, Z80
Little-Endian: address of least significant byte = doubleword address
Motorola 68K, MIPS, IBM 360/370, Xilinx Microblaze, Sparc
Big-Endian: address of most significant byte = doubleword address

Addressing Objects: Endian and Alignment
new ed structures (the "heap" from end, we ignore that here for simplicity)

Note: real compilersplace local variables (the "stack" from beginning of memory,

\[ 1032 = 4 \times 256 + 8 \]
\[ 1001 = 3 \times 256 + 233 \]

```
int x = 258; // 1 \times 256 + 2
char a = "c";
char b = a;
int y = new int[10];
int p = new char[4];

// ASCII = 71
```

- Array: sequence of locations
- Integers: 64 bits (D-word)
- Characters: 8 bits (byte)
- Pointer: address (64 bits)