Review Problem 5

- In assembly, replace the value in X0 with its absolute value.

```
CMP X0, X31
B.GE WAS_NON_NEGATIVE          //B.GE
SUB X0, X31, X0                //X0 = -X0

WAS_NON_NEGATIVE:
```

Loop Example

Compute the sum of the values 0…N-1

```
int sum = 0;
for (int I = 0; I != N; I++) {
        sum += I;
}
```

```
// X0 = N, X1 = sum, X2 = I
    ADD X1, X31, X31 // sum=0
    ADD X2, X31, X31 // I=0
TOP:
    CMP X2, X0       // check I vs N
    B.EQ END         // end when I==N
    ADD X1, X1, X2   // sum += I
    ADDI X2, X2, #1  // I++
    B TOP            // Next iteration
END:
```

```
    ADD X1, X31, X31
    ADD X2, X31, X31
    B TEST
TOP:
    ADD X1, X1, X2
    ADDI X2, X2, #1
TEST:
    CMP X2, X0
    B.NE TOP
```

String toUpper

Convert a string to all upper case

```
char *index = string;
while (*index != 0) { /* C strings end in 0 */
    if (*index >= 'a' && *index <= 'z')
        *index = *index + ('A' - 'a');
    index++;
}
```

('A' - 'a') = -32

```
// string is a pointer held at Memory[80].
// X0=index, 'A' = 65, 'a' = 97, 'z' = 122
    LDUR  X0,[X31,#80]    // index = string
Loop:
    LDURB X1,[X0,#0]      // load byte *index
    CBZ   X1, END         // exit loop if *index == 0
    CMPI  X1,#97          // is *index < 'a'?
    B.LT  NEXT            // don't update if < 'a'
    CMPI  X1,#122         // is *index > 'z'
    B.GT  NEXT            // don't update if > 'z'
    SUBI  X1,X1,#32       // compute *index + ('A' - 'a');
    STURB X1,[X0,#0]      // *index = New value
NEXT:
    ADDI  X0,X0,#1        // index++
    B     Loop            // continue the loop
END:
```

Machine Language vs. Assembly Language

Assembly Language

- mnemonics for easy reading
- labels instead of fixed addresses
- Easier for programmers
- Almost 1-to-1 with machine language

Machine language

- Completely numeric representation
- format CPU actually uses

```
SWAP:
    LSL    X9, X1, #3                                  11010011011 00000 000011 00001 01001
    ADD    X9, X0, X9      // Compute address of v[k]  10001011000 01001 000000 00000 01001
    LDUR   X10, [X9, #0]   // get v[k]                 11111000010 000000000 00 01001 01010
    LDUR   X11, [X9, #8]   // get v[k+1]               11111000010 000001000 00 01001 01011
    STUR   X11, [X9, #0]   // save new value to v[k]   11111000000 000000000 00 01001 01011
    STUR   X10, [X9, #8]   // save new value to v[k+1] 11111000000 000001000 00 01001 01010
    BR     X30             // return from subroutine   11010110000 00000 000000 00000 11110
```