13 – BRANCHING AND LOOPING

COMP256 – COMPUTING ABSTRACTIONS

DICKINSON COLLEGE

MOTIVATING BRANCHES

- Consider the following snippet of HLL code that takes the absolute value of x:

```
if (x < 0) {
  x = -x;
}
```

- What do we need in order to be able to translate this into assembly language?

A BRANCHING EXAMPLE

```
//HLL
int x;

Read X


if (x < 0) {


  x = -x;


}

Print X
```

```
* Assembly Language
X:      .word

        LOAD R0 STDIN
        STORE R0 X

        LOAD R1 #0
        BGEQ R0 R1 NOTNEG    * X >= 0?

        * MM[X] ← -MM[X]
        NOT R0 R0            * X ← ~X
        ADD R0 R0 #1         * X ← X + 1
        STORE R0 X

NOTNEG: STORE R0 STDOUT
        * Continue code here.
```

A BRANCHING EXAMPLE

```
//HLL
int x;

Read X


if (x < 0) {


  x = -x;


}

Print X
```

```
* Assembly Language
X:      .word

        LOAD R0 STDIN
        STORE R0 X

        LOAD R1 #0
        BGEQ R0 R1 NOTNEG    * X >= 0?

        * MM[X] ← -MM[X]
        NOT R0 R0            * X ← ~X
        ADD R0 R0 #1         * X ← X + 1
        STORE R0 X

NOTNEG: STORE R0 STDOUT
        * Continue code here.
```

Notice that branch conditions are often inverted from the HLL when implemented in assembly language.

Labels are used to identify the address for a branch.

THE IF STRUCTURE

```
//HLL

…

if (condition) {

   … if body …
}

…
```

```
* Assembly Language

        …

        conditional branch to ENDIF using the
        inversion of condition

        … if body …

ENDIF:  …
```

BRANCHING INSTRUCTIONS

Instruction	Example	Meaning
JUMP L	JUMP JLOC	PC = JLOC
BNEG R L	BNEG R1 BLOC	IF R1 < 0 THEN PC = BLOC
BPOS R L	BPOS R1 BLOC	IF R1 > 0 THEN PC = BLOC
BZERO R L	BZERO R1 BLOC	IF R1 == 0 THEN PC = BLOC
BNZERO R L	BNZERO R1 BLOC	IF R1 != 0 THEN PC = BLOC
BODD R L	BODD R1 BLOC	IF R1 % 2 != 0 THEN PC = BLOC
BEVEN R L	BEVEN R1 BLOC	IF R1 % 2 == 0 THEN PC = BLOC
BEQ R R L	BEQ R1 R2 BLOC	IF R1 == R2 THEN PC = BLOC
BNEQ R R L	BNEQ R1 R2 BLOC	IF R1 != R2 THEN PC = BLOC
BGEQ R R L	BGEQ R1 R2 BLOC	IF R1 >= R2 THEN PC = BLOC
BLEQ R R L	BLEQ R1 R2 BLOC	IF R1 <= R2 THEN PC = BLOC
BGT R R L	BGT R1 R2 BLOC	IF R1 > R2 THEN PC = BLOC
BLT R R L	BLT R1 R2 BLOC	IF R1 < R2 THEN PC = BLOC

THE IF / ELSE STRUCTURE

```
//HLL

…

if (condition) {

   … if body …

}
else {
   … else body …
}

…
```

```
* Assembly Language

        …

        conditional branch to ELSE using the
        inversion of condition

        … if body …
        JUMP ENDIF

ELSE:   … else body …

ENDIF:  …
```

THE WHILE LOOP STRUCTRUE

```
//HLL

…

while (condition) {

   … loop body …

}

…
```

```
* Assembly Language

        …

        JUMP COND
TOP:    … loop body …


COND:   conditional branch to TOP if
        condition is true.

        …
```

THE FOR LOOP STRUCTURE

```
//HLL

…

for (init; condition; update) {

   … loop body …


}

…
```

```
* Assembly Language

          …

          … init …
          JUMP COND
TOP:      … loop body …

          … update …

COND:     conditional branch to TOP
          if condition is true.

          …
```

ACTIVITY

- Translate (i.e. compile) the following HLL program into assembly language for our machine.

```
//HLL
int max;
int sum=0;

Read max

for (i=1; i<=max; i++) {
  sum = sum + i;
}

Print sum
```

```
MAX:        .word
SUM:        .word 0

            LOAD R0 STDIN      * Read max
            STORE R0 MAX

            LOAD R1 #1         * ...init...        R1 is i
            LOAD R2 SUM        * R2 is temporary sum
            JUMP ENDFOR

FORTOP:     ADD R2 R2 R1       * ...loop body...   sum = sum + i
            ADD R1 R1 #1       * ...update...      i++

ENDFOR:     BLEQ R1 R0 FORTOP * i <= max

            STORE R2 SUM
            STORE R2 STDOUT

            HALT
```