CS 252
Lecture 17; 2015 Oct 19th; Transcribed Lecture notes

Outline

Finish in-class worksheet question 4.

Encoding

In-class worksheet

Write assembly language code to count the number of 1s in the 2’s complement representation of a number. You can assume the number has been loaded into r30.

; revised answer
; r30 is used as program input and shift register
; r31 is used as 1’s counter
; r28 is down counter, done with all 8 bits when r28 hits 0
; r29 is used for andi to check the LSB
; r27 is used to initialize DDRD port for output
ldi r30, 116 ; program input
ldi r31, 0 ; initial 1’s counter
ldi r28, 8 ; initialize down counter

topOfLoop: ; use this label instead of cutting/pasting 8x
mov r29,r30 ; move input to r29
; use a separate register or else andi will destroyed input
andi r29, 1 ; andi with 1, to check the LSB
breq iszero ; jump ahead if zero
subi r31,-1 ; else if LSB is 1, then increment 1’s counter

iszero: ; label to jump to if LSB is 0
asr r30 ; shift right to check the next bit
subi r28,1 ; decrement down counter
breq endofprogram ; go print 1’s counter if done with all 8 bits
rjmp topOfLoop ; loop to check next bit

endofprogram: ; label to start printing
ldi r27,255 ; load all high bits into r27
out 17,r27 ; set DDRD for output
out 18,r31 ; display 1’s counter to output LCD
Cookbook:

**to initialize stack:**

```plaintext
; stack pointer = sp = 256*IO[62] + IO[61]
; SPH = sp / 256
; SPL = sp % 256
; e.g. to set sp to 2436
; SPH = sp / 256 = 2436 / 256 = 9
; SPL = sp % 256 = 2436 % 256 = 132
ldi r16, 9
out 62, r16 ; set SPH
ldi r16, 132
out 61, r16 ; set SPL
```

**to output:**

```plaintext
; output is done by setting DDRD (IO[17]) to all high
; and then writing to PORTD (IO[18])
; e.g. to write 116 to output LCD
ldi r16, 255 ; set all high bits
out 17, r16 ; set DDRD to all high for output
ldi r16, 116 ; load value to display
out 18, r16 ; push value out via PORTD (IO[18])
; you should be able see 116 in simple I/O
; you have to run program first, then change view
```

**to read input:**

```plaintext
; input is done by setting DDRD (IO[17]) to all low
; then using toggle switches to select input
; and finally reading the input via PIND (IO[16])
; e.g. to read in 5 from toggle switches
; first run program
; change view to simple I/O
; toggle pin 0 and pin 2 (which equals 5)
ldi r16, 0 ; set all low bits
out 17, r16 ; set DDRD to all low for input
in r16, 16 ; pull value in via PIND (IO[16])
; you should be able see the value of 5 loaded into r16
```
Encoding

<table>
<thead>
<tr>
<th>required information</th>
<th>instructions requiring this information</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-bit register, 8-bit immediate</td>
<td></td>
</tr>
</tbody>
</table>

All of these instruction can be represented in bits. After getting the mapping, we can build a machine to execute these instructions.