Judah Ben-Eliezer

112352727
9/30/2020

Lab 4:

Logic and Bit Manipulation

Questions:

1.
```
ldi r16, 32        ; decimal
ldi r16, 0x20      ; hex
ldi r16, $20       ; hex
ldi r16, 040       ; octal
ldi r16, 0b00100000    ; binary
```
2. Cp compares the values of two registers. The result is stored in flags in the status register. Cp is typically used for conditional branching, so you can execute different instructions depending on register values.
3. Read-modify-write reads the output values and updates them based on changes. This is particularly useful when using an output register for several different purposes, and updating the value is important.
4. Brge and brlt incorporate the sign of the numbers, while brsh and brlo compare the unsigned values. When working with unsigned values, brsh and brlo are used to avoid interpretation of being negative for 8-bit unsigned values.
5. Active low means an output of 0 from the microcontroller results in a circuit being turned on. In Task 2 this is done by connecting the diode to VCC via a pull up resistor, so that only a 0 output from the microcontroller results in a voltage difference across the diode.
6. Breq takes two cycles when the condition is satisfied, ie. Z = 1 in the status register. This is because the program jumps to a specified branch and this operation takes an extra clock cycle.
7. Sbi would be the preferred method because the out instruction requires first loading a register with the value 0xff
```
Sbi VPORTD_DIR vs ldi r16, 0xff
                  Out VPORTD_DIR, r16
```
8. Common cathode. A resistor network cannot be used because the power input to the LED is unique to each anode and each anode needs its own resistor.
9. Assignment of inputs and outputs to ports in Task 3 is simple, because the input bit position corresponds to the output bit position. Thus, you can simply output the input values for the full color LED and output the complement of the input values for the 3 active low LEDs.
10. No. Because the DED has a common cathode, it is impossible to create different outputs to the cathode. Unless of course you add additional transistors to the circuit in which case an output of 0 from the microcontroller could enable a 1 to a LED anode.

```

AVRASM ver. 2.2.7  E:\ESE_280\$MyDocuments$\Atmel Studio\7.0\lab_4
  \color_computer\color_computer\main.asm Tue Sep 29 18:49:53 2020

E:\ESE_280\$MyDocuments$\Atmel Studio\7.0\lab_4\color_computer\color_computer
  \main.asm(9): Including file 'C:/Program Files (x86)\Atmel\Studio\7.0\Packs
  \atmel\ATmega_DFP\1.2.209\avrasm\inc\m4809def.inc'
E:\ESE_280\$MyDocuments$\Atmel Studio\7.0\lab_4\color_computer\color_computer
  \main.asm(9): Including file 'C:/Program Files (x86)\Atmel\Studio\7.0\Packs
  \atmel\ATmega_DFP\1.2.209\avrasm\inc\m4809def.inc'


                                 ; color_computer.asm
                                 ;
                                 ; Created: 9/23/2020 11:00:26 AM
                                 ; Author : user38x
                                 ;

                                 .list


                                 ; Replace with your application code
                                 start:
000000 e000                         ldi r16, 0x00
000001 ef1f                         ldi r17, 0xFF
000002 b900                          out VPORTA_DIR, r16
000003 b91c                         out VPORTD_DIR, r17
000004 b91d                         out VPORTD_OUT, r17
000005 b918                         out VPORTC_DIR, r17
000006 b909                         out VPORTC_OUT, r16

                                 main:
000007 b102                         in r16, VPORTA_IN
000008 b909                         out VPORTC_OUT, r16
000009 9500                         com r16
00000a b90d                         out VPORTD_OUT, r16
00000b cffb                         rjmp main


RESOURCE USE INFORMATION
------------------------

Notice:
The register and instruction counts are symbol table hit counts,
and hence implicitly used resources are not counted, eg, the
'lpm' instruction without operands implicitly uses r0 and z,
none of which are counted.

x,y,z are separate entities in the symbol table and are
```

```
counted separately from r26..r31 here.

.dseg memory usage only counts static data declared with .byte

"ATmega4809" register use summary:
x  :   0 y  :   0 z  :   0 r0 :   0 r1 :   0 r2 :   0 r3 :   0 r4 :   0
r5 :   0 r6 :   0 r7 :   0 r8 :   0 r9 :   0 r10:   0 r11:   0 r12:   0
r13:   0 r14:   0 r15:   0 r16:   7 r17:   4 r18:   0 r19:   0 r20:   0
r21:   0 r22:   0 r23:   0 r24:   0 r25:   0 r26:   0 r27:   0 r28:   0
r29:   0 r30:   0 r31:   0
Registers used: 2 out of 35 (5.7%)

"ATmega4809" instruction use summary:
.lds  :   0 .sts  :   0 adc   :   0 add   :   0 adiw  :   0 and   :   0
andi  :   0 asr   :   0 bclr  :   0 bld   :   0 brbc  :   0 brbs  :   0
brcc  :   0 brcs  :   0 break :   0 breq  :   0 brge  :   0 brhc  :   0
brhs  :   0 brid  :   0 brie  :   0 brlo  :   0 brlt  :   0 brmi  :   0
brne  :   0 brpl  :   0 brsh  :   0 brtc  :   0 brts  :   0 brvc  :   0
brvs  :   0 bset  :   0 bst   :   0 call  :   0 cbi   :   0 cbr   :   0
clc   :   0 clh   :   0 cli   :   0 cln   :   0 clr   :   0 cls   :   0
clt   :   0 clv   :   0 clz   :   0 com   :   1 cp    :   0 cpc   :   0
cpi   :   0 cpse  :   0 dec   :   0 des   :   0 eor   :   0 fmul  :   0
fmuls :   0 fmulsu:   0 icall :   0 ijmp  :   0 in    :   1 inc   :   0
jmp   :   0 ld    :   0 ldd   :   0 ldi   :   2 lds   :   0 lpm   :   0
lsl   :   0 lsr   :   0 mov   :   0 movw  :   0 mul   :   0 muls  :   0
mulsu :   0 neg   :   0 nop   :   0 or    :   0 ori   :   0 out   :   7
pop   :   0 push  :   0 rcall :   0 ret   :   0 reti  :   0 rjmp  :   1
rol   :   0 ror   :   0 sbc   :   0 sbci  :   0 sbi   :   0 sbic  :   0
sbis  :   0 sbiw  :   0 sbr   :   0 sbrc  :   0 sbrs  :   0 sec   :   0
seh   :   0 sei   :   0 sen   :   0 ser   :   0 ses   :   0 set   :   0
sev   :   0 sez   :   0 sleep :   0 spm   :   0 st    :   0 std   :   0
sts   :   0 sub   :   0 subi  :   0 swap  :   0 tst   :   0 wdr   :   0

Instructions used: 5 out of 114 (4.4%)

"ATmega4809" memory use summary [bytes]:
Segment   Begin    End      Code   Data   Used    Size   Use%
---------------------------------------------------------------
[.cseg] 0x000000 0x000018     24      0     24   49152   0.0%
[.dseg] 0x002800 0x002800      0      0      0    6144   0.0%
[.eseg] 0x000000 0x000000      0      0      0     256   0.0%

Assembly complete, 0 errors, 0 warnings

```