```
AVRASM ver. 2.1.52  C:\Users\radra_000\Box Sync\college sophomore fall 2014\fall 2014 notes and files
\ese 380 lab\lab 6\nibble_load\nibble_load\nibble_load.asm Fri Oct 10 15:30:50 2014

C:\Users\radra_000\Box Sync\college sophomore fall 2014\fall 2014 notes and files\ese 380 lab\lab 6\
nibble_load\nibble_load\nibble_load.asm(21): Including file 'C:\Program Files (x86)\Atmel\Atmel Tool
chain\AVR Assembler\Native\2.1.39.1005\avrassembler\Include\m16def.inc'


                 * nibble_load.asm
                 *
                 ;This program will wait for the press of pushbutton 1 connected to PC0,
                 ;when pressed, will load the values of the lower nibble dip switch,
                 ; and display the the output in 7seg display. anything above nine will
                 ;be ignored and displayed as zero
                 ;
                 ;debouncing will not be required because,no matter how many times the dip
                 ;switch is read due to bounce, the input value remain constant for the
                 ;bounce.
                 ;
                 ;Inputs - dip switches conected to Port D, pbsw connected to PC0
                 ;outputs - 7seg dispay connecte to port B
                 ;

                 *  Created: 10/9/2014 10:16:05 AM
                 *   Author: radra_000
                 */
                 .list

                reset:
000000 ef0f         ldi r16, $ff                    ;set port a and port b
000001 bb07         out ddrb, r16                   ;into outputs
                // out ddra, r16                    ;by loading 1s to the data direction register
000002 bb02         out portd, r16                  ;enable pullup resistors for the dip switch
000003 9aa8         sbi portc, 0                    ;enable pullup resistor for the pushbutton1
000004 e076         ldi r23, 6                      ;load r23 with 6 to check weather input is>10
000005 e000         ldi r16, $00                    ;set port c and port d
000006 bb04         out ddrc, r16                   ;into inputs
000007 bb01         out ddrd, r16                   ;by loading 0s into the data direction register
                    //ldi r18, $00

                main_loop:
000008 9998         sbic pinc, 0                    ;check if LOAD pushbutton is pressed
000009 cffe         rjmp main_loop                  ;if not then check again
00000a b312         in r17, portD                   ;take value of portD into r17
00000b 701f         andi r17, $0F                   ;force the upper nibbles to 0
00000c 2f21         mov r18, r17                    ;copy value of r17 to r18
00000d 0f27         add r18, r23                        ;to check if greater than 9
00000e f008         brcs check                      ;if greater, go to check
00000f c001         rjmp hex_7seg                   ;jump to hex7seg otherwise

                //check will output 0 in the 7seg if the dip switch value is
                //greater than 9
                check:
000010 e010         ldi r17, 0                      ;load r17 with 0 to diplay 0

                hex_7seg:
                    //mov r17, r18                  ;copy r18 to r17
000011 e0f0         ldi ZH, HIGH(table*2)
000012 e3e6         ldi ZL, LOW(table*2)            ;set z to point to start of the table
000013 e000         ldi r16, $00                    ;clear for later use
000014 0fe1         add ZL, r17                     ;add low byte
000015 1ff0         adc ZH, r16                     ;add in the carry
000016 9114         lpm r17, z                      ;load bid pattern from table into r18
                display:
000017 bb18         out PORTB,r17                   ;output patter for 7 seg display
```

```
                wait_1:
000018 9b98        sbis pinc, 0                    ;wait for a logic 1(when pushbutton is released)
000019 cffe        rjmp wait_1                     ;before updating the values
00001a cfed        rjmp main_loop
00001b 7940
00001c 3024
00001d 1219
00001e 7803
00001f 1800     table: .db $40, $79, $24, $30, $19, $12, $03, $78,$0, $18
                       //  0    1     2   3    4     5    6    7  8   9
                /*
                delay:
                   ldi r18,100
                   outer:
                       ldi r19 33
                       inner:
                           dec r19
                           brne inner
                           dec r18
                           brne outer
                   ret



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

"ATmega16" register use summary:
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:   8 r17:   7 r18:   2 r19:   0 r20:   0 r21:   0 r22:   0 r23:   2
r24:   0 r25:   0 r26:   0 r27:   0 r28:   0 r29:   0 r30:   2 r31:   2
x  :   0 y  :   0 z  :   1
Registers used: 7 out of 35 (20.0%)

"ATmega16" instruction use summary:
.lds  :   0 .sts  :   0 adc   :   1 add   :   2 adiw  :   0 and   :   0
andi  :   1 asr   :   0 bclr  :   0 bld   :   0 brbc  :   0 brbs  :   0
brcc  :   0 brcs  :   1 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   :   0 cp    :   0 cpc   :   0
cpi   :   0 cpse  :   0 dec   :   0 eor   :   0 fmul  :   0 fmuls :   0
fmulsu:   0 icall :   0 ijmp  :   0 in    :   1 inc   :   0 jmp   :   0
ld    :   0 ldd   :   0 ldi   :   7 lds   :   0 lpm   :   2 lsl   :   0
lsr   :   0 mov   :   1 movw  :   0 mul   :   0 muls  :   0 mulsu :   0
neg   :   0 nop   :   0 or    :   0 ori   :   0 out   :   5 pop   :   0
push  :   0 rcall :   0 ret   :   0 reti  :   0 rjmp  :   4 rol   :   0
ror   :   0 sbc   :   0 sbci  :   0 sbi   :   1 sbic  :   1 sbis  :   1
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: 13 out of 113 (11.5%)

"ATmega16" memory use summary [bytes]:
Segment   Begin    End      Code   Data   Used    Size   Use%
----------------------------------------------------------------
[.cseg] 0x000000 0x000040     54     10     64   16384   0.4%
[.dseg] 0x000060 0x000060      0      0      0    1024   0.0%
[.eseg] 0x000000 0x000000      0      0      0     512   0.0%

Assembly complete, 0 errors, 0 warnings
```