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\cond_trans_select_srff\cond_trans_select_srff\cond_trans_select_srff.asm Tue Oct 14 20:22:37 2014

C:\Users\radra_000\Box Sync\college sophomore fall 2014\fall 2014 notes and files\ese 380 lab\lab 6\cond_trans_select_srff\cond_trans_select_srff\cond_trans_select_srff.asm(25): Including file 'C:\Program Files (x86)\Atmel\Atmel Toolchain\AVR Assembler\Native\2.1.39.1005\avrassembler\Include\m16def.inc'

```
                 * cond_trans_select_srff.asm
                 *
                 ; This program is a modification of the second program and
                 ; will utilize 2 pbsw, pbsw 3 will used with the d_ff and act
                 ; as select and Pbsw2(pc6) will be used as load. When the load
                 ; press is recognized a 1 will be outputted to PA6 to reset the
                 ; input going into PA7. Thus acting as a srff. Two leds will
                 ; be attached to PA0 and PA1. PA0 led will be turned on when the
                 ; lower nibble dip switch is used, and PA1 led for the other.
                 ; select pbsw will alternate between lower and upper dip switch
                 ; for each press. load will load the input values of the selected
                 ; dip switch.
                 ;
                 ;inputs : dip switch (Port D), PBSW 3 and 2(PC7,PC6)
                 ;outputs: 7seg display (Port B), led 1 and 2(PA0,PA1),
                 ;         r18 used for porta output
                 ;         r17 used to store and output dip switch input

                 *  Created: 10/9/2014 10:53:35 AM
                 *   Author: raymond ng
                 */

                 .list

                reset:
                   //initizling the stack pointer
000000 e50f        ldi r16, LOW(RAMEND)           ;load SPL with low byte of
000001 bf0d        out SPL, r16                   ;RAMEND adress
000002 e004        ldi r16, HIGH(RAMEND)          ;load SPH with low byte of
000003 bf0e        out SPH, r16                   ;RAMEND adress
000004 ef0f        ldi r16, $FF                   ;load r16 with 1's and
000005 bb02        out portd, r16                 ;turn on pull up resistors in portd
000006 bb07        out ddrb, r16                  ;make portb as output
000007 e403        ldi r16, $43                   ;PC6 and PC1, PC0 as output
000008 bb0a        out ddra, r16                  ;port a as output
000009 9aa8        sbi portc, 0                   ;turn on pull up resistors in PC0
00000a 9aae        sbi portc, 6                   ;turn on pull ups in pc6
00000b e000        ldi r16, $00                   ;load r16 with 0's
00000c bb01        out ddrd, r16                  ;set portd and
00000d bb04        out ddrc, r16                  ;port c as inputs
00000e e010        ldi r17, $00                   ;load r17 with 0's
00000f e021        ldi r18, $01                   ;load r18 with 0's
000010 e30f        ldi r16, $3F                   ;output "-" in 7seg
000011 bb07        out ddrb, r16                  ;indicating no input
000012 9ac8        sbi pina, 0                    ;indicate lower nibble
000013 98c9        cbi pina, 1                    ;as default
000014 9ace        sbi pina, 6                    ;set default clear to be 1
                main_loop:
000015 b312        in r17, portd                  ;input values of dip switch
000016 99cf        sbic pina, 7                   ;wait for load button press
000017 c00a        rjmp LOAD_push                 ;check for other push button
000018 d019        rcall delay                    ;jump
000019 9b9f        sbis pinc,7                    ;back to
00001a c005        rjmp clear                     ;main loop if false
00001b 9512        swap r17                       ;swap nibbles
00001c e003        ldi r16, $03                   ;set 00000011
```

```
00001d b32b         in r18, porta                   ;take porta values
00001e 2720         eor r18, r16                    ;toggle first two bits
00001f bb2b         out porta, r18                  ;turn on the led 1 or 2;
                    clear:
000020 98ce         cbi pina, 6                     ;generate pulse
000021 9ace         sbi pina, 6                     ;to clear register
                LOAD_push:
000022 999e         sbic pinc, 6                    ;check for load
000023 cff1         rjmp main_loop                  ;pushbutton
000024 701f         andi r17, $0F                   ;force upper values to 0
                hex_7seg:
                    //mov r17, r18                  ;copy r18 to r17
000025 e0f0         ldi ZH, HIGH(table*2)
000026 e5ea         ldi ZL, LOW(table*2)            ;set z to point to start of the table
000027 e000         ldi r16, $00                    ;clear for later use
000028 0fe1         add ZL, r17                     ;add low byte
000029 1ff0         adc ZH, r16                     ;add in the carry
00002a 9114         lpm r17, z                      ;load bid pattern from table into r18
                display:
00002b bb18         out PORTB,r17                   ;output patter for 7 seg display
00002c cfe8         rjmp main_loop
00002d 7940
00002e 3024
00002f 1219
000030 7803
000031 1800     table: .db $40, $79, $24, $30, $19, $12, $03, $78, $0, $18
                       //  0    1     2   3    4    5    6    7    8   9
                /*
                ;  Delay:
                ;      This subroutine will utilize variables r16, and r17.
                ;      The variabled will be initialized to act as counters
                ;      to count a 10 ms delay
                ;      r17: 100, r16: 33
                */


                delay:
000032 931f         push r17                        ;push aside the registers
000033 930f         push r16                        ;r16 and r17
000034 e201         ldi r16, 33                     ;loop through 33 times
                    outer:
000035 e614         ldi r17, 100                    ;of 100 decrements
                    inner:
000036 951a         dec r17
000037 f7f1         brne inner
000038 950a         dec r16
000039 f7d9         brne outer
00003a 910f         pop r16                         ;pop the registers back
00003b 911f         pop r17
00003c 9508         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:  22 r17:  11 r18:   4 r19:   0 r20:   0 r21:   0 r22:   0 r23:   0
r24:   0 r25:   0 r26:   0 r27:   0 r28:   0 r29:   0 r30:   2 r31:   2
x  :   0 y  :   0 z  :   1
Registers used: 6 out of 35 (17.1%)

"ATmega16" instruction use summary:
.lds  :   0 .sts  :   0 adc   :   1 add   :   1 adiw  :   0 and   :   0
andi  :   1 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  :   2 brpl  :   0 brsh  :   0 brtc  :   0 brts  :   0 brvc  :   0
brvs  :   0 bset  :   0 bst   :   0 call  :   0 cbi   :   2 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   :   2 eor   :   1 fmul  :   0 fmuls :   0
fmulsu:   0 icall :   0 ijmp  :   0 in    :   2 inc   :   0 jmp   :   0
ld    :   0 ldd   :   0 ldi   :  14 lds   :   0 lpm   :   2 lsl   :   0
lsr   :   0 mov   :   0 movw  :   0 mul   :   0 muls  :   0 mulsu :   0
neg   :   0 nop   :   0 or    :   0 ori   :   0 out   :  10 pop   :   2
push  :   2 rcall :   1 ret   :   1 reti  :   0 rjmp  :   4 rol   :   0
ror   :   0 sbc   :   0 sbci  :   0 sbi   :   5 sbic  :   2 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  :   1 tst   :   0 wdr   :   0
Instructions used: 20 out of 113 (17.7%)

"ATmega16" memory use summary [bytes]:
Segment   Begin    End      Code   Data   Used    Size   Use%
----------------------------------------------------------------
[.cseg] 0x000000 0x00007a    112     10    122   16384   0.7%
[.dseg] 0x000060 0x000060      0      0      0    1024   0.0%
[.eseg] 0x000000 0x000000      0      0      0     512   0.0%

Assembly complete, 0 errors, 0 warnings
```