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_select_rajith\cond_select_rajith\cond_select_rajith.asm
   Fri Oct 10 15:26:47 2014

C:\Users\radra_000\Box Sync\college sophomore fall 2014\fall 2014 notes and files\ese 380 lab
\lab 6\cond_select_rajith\cond_select_rajith\cond_select_rajith.asm(26): Including file '
C:\Program Files (x86)\Atmel\Atmel Toolchain\AVR Assembler\Native\2.1.39.1005\avrassembler
\Include\m16def.inc'

```
                  * cond_trans_select.asm
                  *
                  ; This program will utilize 2 pbsw, pbsw 1(pc0) will will used
                  ; as select and Pbsw2(pc6) will be used as load. 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 1 and 2(PC0,PC6).
                  ;outputs: 7seg display (Port B), led 1 and 2(PA0,PA1)
                  ;r17- stores dip switch values
                  ;r18 - eliminates upper or lower nible depending on the select
                  ;r19 - alternates between 01 and 10 to turn on the leds
                  ;r20 - serves as a check for value above 9 and used in delay
                  ;      subroutines with value 100
                  ;r21 - has a value of 33, used in delay loop, combined with
                  ;      r21 and r20 will delay for 9999ms
                  *  Created: 10/9/2014 10:53:35 AM
                  *   Author: radra_000
                  */

                  .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 bb07         out ddrb, r16                   ;make portb as output
000006 bb0a         out ddra, r16                   ;port a as output
000007 bb02         out portd, r16                  ;turn on pull up resistors in portd
000008 9aa8         sbi portc, 0                    ;turn on pull up resistors in PC0
000009 9aae         sbi portc, 6                    ;turn on pull ups in pc6
00000a e000         ldi r16, $00                    ;load r16 with 0's
00000b bb01         out ddrd, r16                   ;set portd and
00000c bb04         out ddrc, r16                   ;port c as inputs
00000d e010         ldi r17, $00                    ;load r17 with 0's
00000e ef20         ldi r18, $F0                    ;load r18 with 11110000 to read
                                                    ; upper or lower switch
00000f e076         ldi r23, 6                      ;to check wheather or not input value
                                                    ;is <10
000010 e031         ldi r19, 0b01                   ;load r19 with 01 to turn on led
000011 e30f         ldi r16, $3F                    ;output "-" in 7seg
000012 bb07         out ddrb, r16                   ;indicating no input
                  main_loop:

000013 999e         sbic pinc, 6                    ;wait for load button press
000014 c009         rjmp check_LOAD                 ;go to check load if pressed
000015 9998         sbic pinc, 0                    ;wait for select button press
000016 c001         rjmp check_select               ;go to checkselect if pressed
000017 cffb         rjmp main_loop                  ;repeat the code
```

```
                ;when called will alternate between reading upper nibble and lower
                ;nibble for every press of the select switch
                check_select:
000018 d029        rcall delay                    ;delay 10ms for debounce
000019 9b98        sbis pinc,0                    ;check if select is still pressed
00001a cff8        rjmp main_loop                 ;if not pressed go to main loop
00001b b310        in r17, pind                   ;input values of dip switch
00001c d019        rcall selectingnibble          ;go to selectingnibble subroutine
00001d cff5        rjmp main_loop                 ;go to main loop

                ;when load button is pressed, will take the current value of r17 and sends to
                ;hex7seg subroutine
                check_load:
00001e d023        rcall delay                    ;delay 10ms debounce
00001f 9b9e        sbis pinc, 6                   ;check if load is  still pressed
000020 cff2        rjmp main_loop                 ;if not pressed go to main loop
000021 2f49        mov r20, r25                   ;copy bits from r25 to r20 to check
000022 0f47        add r20, r23                   ;wheather is above 9
000023 f010        brcs dis_zero                  ;if above 9 display zero
000024 d004        rcall hex_7seg                 ;go to hex7seg and display the
                                                  ;value in 7seg
000025 cfed        rjmp main_loop                 ;go to main loop

                ;when called will display 0 in the 7seg display
                dis_zero:
000026 e090        ldi r25, 0                     ;load r25 with0
000027 d001        rcall hex_7seg                 ;go to hex7seg and display the
                                                  ;value in 7seg
000028 cfea        rjmp main_loop                 ;go to main loop

                ;when called, will take the value in r17 and diplays it in the 7seg
                hex_7seg:
000029 e0f0        ldi ZH, HIGH(table*2)
00002a e6e2        ldi ZL, LOW(table*2)           ;set z to point to start of the table
00002b e000        ldi r16, $00                   ;clear for later use
00002c 0fe9        add ZL, r25                    ;add low byte
00002d 1ff0        adc ZH, r16                    ;add in the carry
00002e 9194        lpm r25, z                     ;load bid pattern from table into r25
                display:
00002f bb98        out PORTB,r25                  ;output patter for 7 seg display
000030 9508        ret
000031 7940
000032 3024
000033 1219
000034 7803
000035 1800        table: .db $40, $79, $24, $30, $19, $12, $03, $78,$0, $18
                          //  0    1     2    3    4    5    6    7 8   9

                selectingnibble:
000036 2f91        mov r25, r17                   ;copy r17 to r25
000037 9520        com r18                        ;com r17, to alternate between the
000038 2392        and r25, r18                   ;upper nibble and lower nibble
000039 9530        com r19                        ;turn led upper or lower
00003a fd30        sbrc r19, 0                    ;skip if bit 0 is 0, indicating upper nibble
00003b d002        rcall swap_nibble              ; goto swap nibble to swap the upper to lower
00003c d003        rcall dis_led                  ;display the led to indicate
00003d 9508        ret

                ;when the r19 is 10, indicating the upper nibble is selected
                ;the digits is r17 will be swapped, so the 7seg could be display1
                swap_nibble:
00003e 9592        swap r25                       ;swap r17, so the upper nibble will be in
                                                  ;lower nibble
00003f 9508        ret

                //diplay the corresponding led to nibble,
```

```
                    ;when r19:10 the upper nibble led will be on
                    ;when r19:01 the lower nibble led will be on
                    dis_LED:
                        //code to yet be determined based on led placements
000040 bb3b             out porta, r19                 ;turn on the led 1 or 2;
000041 9508             ret


                    ;delays for 10ms

                    delay:
000042 e644             ldi r20,100
                        outer:
000043 e251                 ldi r21, 33
                            inner:
000044 955a                     dec r21
000045 f7f1                     brne inner
000046 954a                     dec r20
000047 f7d9                     brne outer
000048 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:  15 r17:   3 r18:   3 r19:   4 r20:   4 r21:   2 r22:   0 r23:   2
r24:   0 r25:   8 r26:   0 r27:   0 r28:   0 r29:   0 r30:   2 r31:   2
x  :   0 y  :   0 z  :   1
```
Registers used: 11 out of 35 (31.4%)

"ATmega16" instruction use summary:
```
.lds  :   0 .sts  :   0 adc   :   1 add   :   2 adiw  :   0 and   :   1
andi  :   0 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  :   2 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   :   2 cp    :   0 cpc   :   0
cpi   :   0 cpse  :   0 dec   :   2 eor   :   0 fmul  :   0 fmuls :   0
fmulsu:   0 icall :   0 ijmp  :   0 in    :   1 inc   :   0 jmp   :   0
ld    :   0 ldd   :   0 ldi   :  15 lds   :   0 lpm   :   2 lsl   :   0
lsr   :   0 mov   :   2 movw  :   0 mul   :   0 muls  :   0 mulsu :   0
neg   :   0 nop   :   0 or    :   0 ori   :   0 out   :  10 pop   :   0
push  :   0 rcall :   7 ret   :   5 reti  :   0 rjmp  :   8 rol   :   0
ror   :   0 sbc   :   0 sbci  :   0 sbi   :   0 sbic  :   2 sbis  :   2
sbiw  :   0 sbr   :   0 sbrc  :   0 sbrs  :   1 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	0x000092	136	10	146	16384	0.9%
[.dseg]	0x000060	0x000060	0	0	0	1024	0.0%
[.eseg]	0x000000	0x000000	0	0	0	512	0.0%

Assembly complete, 0 errors, 0 warnings