```
AVRASM ver. 2.2.7  F:\lab7\lab7part1\labpart3\main.asm Fri Oct 23 15:36:22 2020

F:\lab7\lab7part1\labpart3\main.asm(12): Including file 'C:/Program Files (x86)
  \Atmel\Studio\7.0\Packs\atmel\ATmega_DFP\1.2.209\avrasm\inc\m4809def.inc'
F:\lab7\lab7part1\labpart3\main.asm(12): Including file 'C:/Program Files (x86)
  \Atmel\Studio\7.0\Packs\atmel\ATmega_DFP\1.2.209\avrasm\inc\m4809def.inc'


                                 ; labpart3.asm
                                 ;
                                 ; Created: 10/23/2020 2:29:49 PM
                                 ; Author : user38x
                                 ;


                                 ; Replace with your application code
                                 ; Replace with your application code
                                 .list

                                 ; Replace with your application code
                                     start:
                                  ; configure I/O ports
000000 e000                          ldi r16, 0x00               ;change r17 to
  all 0s for input
000001 b900                         out VPORTA_DIR, r16         ;PORTA - all pins
  configured as inputs
000002 ef1f                         ldi r17, 0xFF               ;load r17 with all
   1s
000003 bb10                         out VPORTE_DIR, r17         ;sets PORTE as an
  output
000004 b91c                         out VPORTD_DIR, r17         ;sets PORTF as an
  output
000005 b91d                          out VPORTD_OUT, r17         ;sets PORTD as
  output

                                    main_loop:
000006 b91d                         out VPORTD_OUT, r17
000007 9b91                         sbis VPORTE_IN, 1      ;checks if flip flop is
   on, button is pushed
000008 cffd                         rjmp main_loop         ;goes back to beginning
   of loop if button released
000009 c000                         rjmp take_in           ;goes to display if
  button pushed

                                    take_in:
00000a e020                         ldi r18, 0x00                     ;sets r18 to
   a blank register
00000b e002                         ldi r16, VPORTA_IN                ;loads r16
```

```
  with switch inputs
00000c d015                    rcall reverse                    ;reverses
  r16
00000d 2f30                    mov r19, r16                    ;moves
  reversed number into new registert to get bits representing display digits
00000e 7033                    andi r19, 0x03                ;ands r19 with
  0000 0011 to get only first two bits
00000f 9506                    lsr r16                      ;shifts r16 4
  times to get only 4 bits
000010 9506                    lsr r16
000011 9506                    lsr r16
000012 9506                    lsr r16
000013 d014                    rcall hex_to_7seg              ;converts hex
  number to 7 segment display pattern
000014 ef4f                    ldi r20, 0xFF                  ;create a
  register representing first digit with everything off at first
000015 2f54                    mov r21, r20                    ;create a
  register representing 2nd digit with everything off at first
000016 2f64                    mov r22, r20                    ;create a
  register representing 3rd digit with everything off at first
000017 2f74                    mov r23, r20                    ;create a
  register representing 4th digit with everything off at first
000018 9a91                    sbi VPORTE_IN , 1              ;clears flip
  flop
000019 3030                    cpi r19, 0x00                    ;check if
  switch is set to first digit
00001a f161                    breq first_digit                   ;branches
  to set first digit if r19 = 0
00001b 3031                    cpi r19, 0x01                      ;check if
   switch is set to 2nd digit
00001c f161                    breq second_digit                   ;branches
   to set 2nd digit if r19 = 1
00001d 3032                    cpi r19, 0x02                       ;check if
    switch is set to 3rd digit
00001e f161                    breq third_digit                    ;branches
   to set 3rd digit if r19 = 2
00001f 3033                    cpi r19, 0x03                      ;check if
   switch is set to 4th digit
000020 f161                    breq fourth_digit                   ;branches
   to set 4th digit if r19 = 3
000021 cfe4                    rjmp main_loop                 ;goes back to
  main loop


;***************************************************************************
                                 ;*
                                 ;* "reverses" - reverses a register
                                 ;*
                                 ;* Description: Reverses a register using two
```

```
                     different registers.
                                 ;* shifts r16 then moving that shifted bit into
                     r17 8 times to reverse
                                 ;*
                                 ;* Author:                    Tyler Ovenden
                                 ;* Version:                       1.0

                                 ;* Last updated:               102120
                                 ;* Target:                     ATmega4809
                                 ;* Number of words:                11
                                 ;* Number of cycles:
                                 ;* Low registers modified:     none
                                 ;* High registers modified:        r16, r17
                                 ;*
                                 ;* Parameters: r16: input from switch
                                 ;* Returns: r16: reversed switch input, shifted 4
                      times to get only bits 7-4 from reversed bit
                                 ;*
                                 ;* Notes:
                                 ;*
                                 ;************************************************
                     ***************************

                                  reverse:
000022 9506                       lsr r16                     ;shifts r16 once
  putting msb in flag
000023 1f11                       rol r17                     ;rotates r17 once
   placing carry bit from lsr into r17
000024 3000                       cpi r16, 0x00               ;checks if r16 is
   all 0
000025 f7e1                        brne reverse                ;if r16 is not 0
   then repeat loop
000026 2f01                       mov r16, r17                ;moves reversed
  number in r17 to r16
000027 9508                       ret                         ;ends subroutine

;****************************************************************************
                                 ;*
                                 ;* "hex_to_7seg" - Hexadecimal to Seven Segment
                     Conversion
                                 ;*
                                 ;* Description: Converts a right justified
                     hexadecimal digit to the seven
                                 ;* segment pattern required to display it.
                     Pattern is right justified a
                                 ;* through g. Pattern uses 0s to turn segments on
                      ON.
                                 ;*
                                 ;* Author:                    Ken Short
```

```
                            ;* Version:                        1.0

                            ;* Last updated:               101620
                            ;* Target:                     ATmega4809
                            ;* Number of words:                8
                            ;* Number of cycles:           13
                            ;* Low registers modified:     none
                            ;* High registers modified:        r16, r18, ZL, 
                  ZH
                            ;*
                            ;* Parameters: r18: right justified hex digit, 
                  high nibble 0
                            ;* Returns: r18: segment values a through g right 
                   justified
                            ;*
                            ;* Notes:
                            ;*
                            ;*************************************************
                  ***************************
                            hex_to_7seg:
000028 702f                    andi r18, 0x0F              ;clear ms nibble
000029 e0f0                     ldi ZH, HIGH(hextable * 2)  ;set Z to point 
  to start of table
00002a e6e0                     ldi ZL, LOW(hextable * 2)
00002b e000                     ldi r16, $00                ;add offset to Z 
  pointer
00002c 0fe2                     add ZL, r18
00002d 1ff0                     adc ZH, r16
00002e 9124                     lpm r18, Z                  ;load byte from 
  table pointed to by Z
00002f 9508                    ret

                                ;Table of segment values to display digits 0 
                  - F
                                ;!!! seven values must be added - verify all 
                  values
000030 4f01
000031 0612
000032 244c
000033 0f20
000034 0400
000035 6008
000036 3231
000037 3830                 hextable: .db $01, $4F, $12, $06, $4C, $24, $20, 
  $0F, $00, $04, $08, $60, $31, $32, $30, $38

                            display:
000038 b94d                 out VPORTD_OUT, r20             ;sets 7 segment 
  display value for first display digit
```

```
000039 e70f                     ldi r16, 0x7F                  ;sets value for 
  transitors to display 1st digit
00003a b909                     out VPORTC_OUT, r16            ;turns on 1st 
  digit
00003b b95d                     out VPORTD_OUT, r21            ;sets 7 segment 
  display value for 2nd display digit
00003c eb0f                     ldi r16, 0xBF                   ;sets value for 
  transitors to display 2nd digit
00003d b909                     out VPORTC_OUT, r16            ;turns on 2nd 
  digit
00003e b96d                     out VPORTD_OUT, r22            ;sets 7 segment 
  display value for 3rd display digit
00003f ed0f                     ldi r16, 0xDF                   ;sets value for 
  transitors to display 3rd digit
000040 b909                     out VPORTC_OUT, r16            ;turns on 3rd 
  digit
000041 b97d                     out VPORTD_OUT, r23            ;sets 7 segment 
  display value for 4th display digit
000042 eb0f                     ldi r16, 0xBF                   ;sets value for 
  transitors to display 4th digit
000043 b909                     out VPORTC_OUT, r16             ;turns on 4th 
  digit
000044 9b91                     sbis VPORTE_IN, 1               ;checks if flip 
  flop output is 1, pushed down
000045 cff2                     rjmp display                    ;loops display 
  if flip flop released
000046 cfc3                     rjmp take_in                    ;restarts loop 
  to take in switch values



                                first_digit:
000047 2f42                     mov r20, r18                    ;places hex 
  digits into register reprsenting 1st display digit
000048 cfef                     rjmp display                    ;calls display 
  for 7 segment display

                                second_digit:
000049 2f52                     mov r21, r18                   ;places hex 
  digits into register reprsenting 2nd display digit
00004a cfed                     rjmp display                   ;calls display 
  for 7 segment display

                                third_digit:
00004b 2f62                     mov r22, r18                   ;places hex 
  digits into register reprsenting 3rd display digit
00004c cfeb                     rjmp display                    ;calls display 
  for 7 segment display

                                fourth_digit:
```

```
00004d 2f72                     mov r23, r18                    ;places hex
  digits into register reprsenting 4th display digit


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  :   1 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:  21 r17:   7 r18:   8 r19:   6 r20:   6
r21:   3 r22:   3 r23:   3 r24:   0 r25:   0 r26:   0 r27:   0 r28:   0
r29:   0 r30:   2 r31:   2
Registers used: 11 out of 35 (31.4%)

"ATmega4809" instruction use summary:
.lds  :   0 .sts  :   0 adc   :   1 add   :   1 adiw  :   0 and   :   0
andi  :   2 asr   :   0 bclr  :   0 bld   :   0 brbc  :   0 brbs  :   0
brcc  :   0 brcs  :   0 break :   0 breq  :   4 brge  :   0 brhc  :   0
brhs  :   0 brid  :   0 brie  :   0 brlo  :   0 brlt  :   0 brmi  :   0
brne  :   1 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   :   5 cpse  :   0 dec   :   0 des   :   0 eor   :   0 fmul  :   0
fmuls :   0 fmulsu:   0 icall :   0 ijmp  :   0 in    :   0 inc   :   0
jmp   :   0 ld    :   0 ldd   :   0 ldi   :  12 lds   :   0 lpm   :   2
lsl   :   0 lsr   :   5 mov   :   9 movw  :   0 mul   :   0 muls  :   0
mulsu :   0 neg   :   0 nop   :   0 or    :   0 ori   :   0 out   :  13
pop   :   0 push  :   0 rcall :   2 ret   :   2 reti  :   0 rjmp  :   9
rol   :   1 ror   :   0 sbc   :   0 sbci  :   0 sbi   :   1 sbic  :   0
sbis  :   2 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: 17 out of 114 (14.9%)
```

```
"ATmega4809" memory use summary [bytes]:
Segment   Begin    End      Code   Data   Used    Size   Use%
---------------------------------------------------------------
[.cseg] 0x000000 0x00009e    142     16    158   49152   0.3%
[.dseg] 0x002800 0x002800      0      0      0    6144   0.0%
[.eseg] 0x000000 0x000000      0      0      0     256   0.0%

Assembly complete, 0 errors, 0 warnings
```