Prelab 4

Author: Rajith Radhakrishnan

ID: 109061463

Partner's Name: Raymond Ng

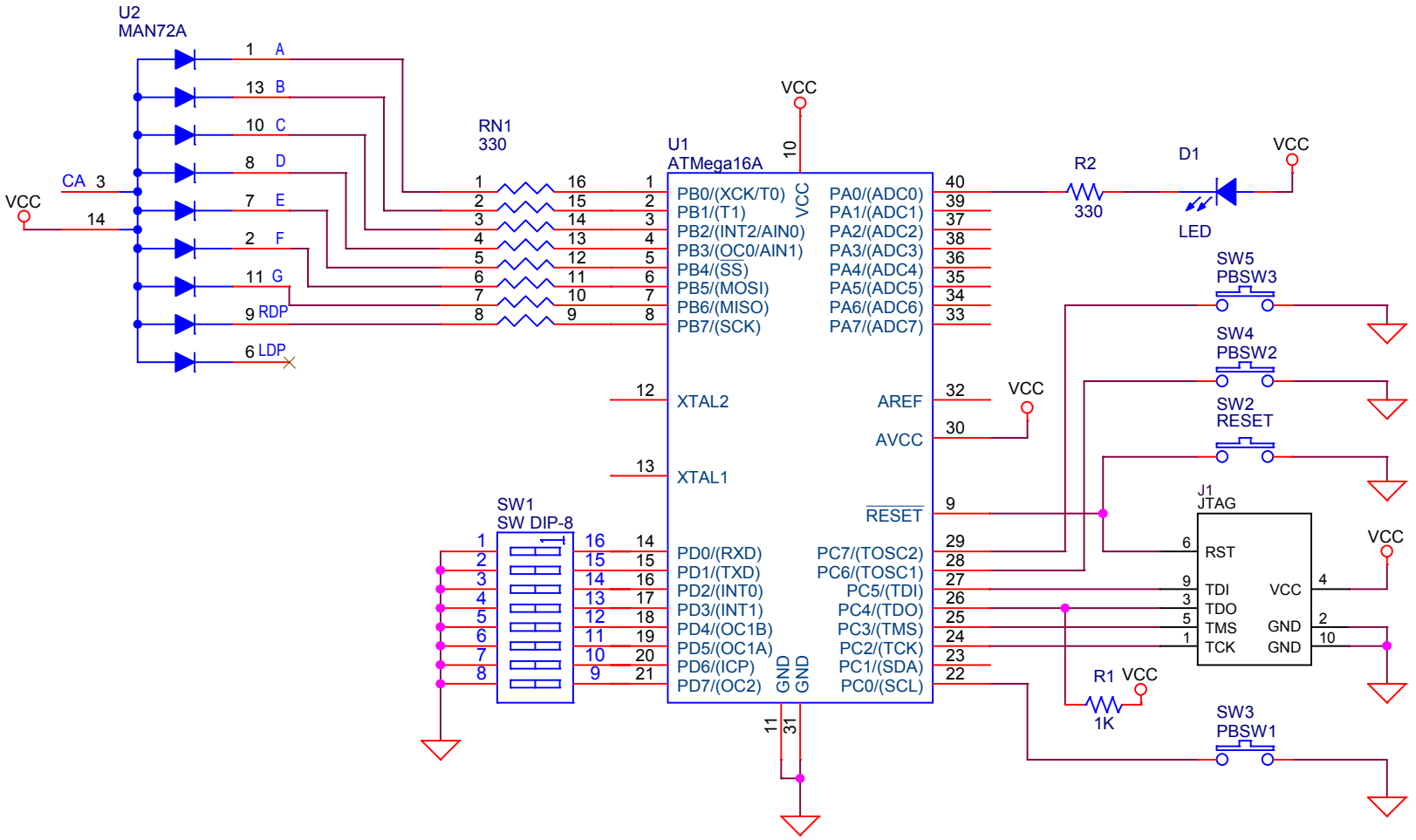
ID: 109223276

Course and section: ESE 380 L01

Bench Number: 6

Due Date/time : 9/30/14 9:00 PM

Date of the Lab: 10/1/14



Title	Lab3 - task 4	
Size A	Document Number Ng, Radhakrishnan	Rev 1V
Date: Monday, September 29, 2014	Sheet 1 of 1	

AVRASM ver. 2.1.52 C:\Users\radra_000\Box Sync\college sophomore fall 2014\fall 2014 notes and files\ese 380 lab\lab 4\7-seg-diag\7-seg-diag\7-seg-diag.asm Tue Sep 30 20:50:05 2014

C:\Users\radra_000\Box Sync\college sophomore fall 2014\fall 2014 notes and files\ese 380 lab\lab 4\7-seg-diag\7-seg-diag\7-seg-diag.asm(24): Including file 'C:\Program Files (x86)\Atmel\Atmel Toolchain\AVR Assembler\Native\2.1.39.1005\avrassembler\Include\m16def.inc'

```
                 *-----------------------------------------------------------
                 * 7-seg-diag.asm
                 ; This program will enable a user to turn on and off all segments
                 ; of the 7seg, with a press of a pushbutton switch (PBSW1).
                 *
                 * Inputs used: PD0 to PD7 (DIP-8 switch)
                 *             PC0, PC6, PC7 (PBSW1, PBSW2, PBSW3,active low)
                 * Outputs used: PB0 to PB7 (7 seg display, active low)
                 *              PA0 (overflow indicating LED)
                 *
                 *
                 * assumes: nothing
                 * alters: r16, SREG
                 *
                 * Author: Rajith Radhakrishnan (109061463) , Raymond Ng(109223276)
                 * Date: 10/01/14
                 * ESE 380 L01, Bench 6
                 * Version 1.0
                 */


                 .list

                 //load all the pins in port B, Port D and port A 0, as outputs, Port C (0,6,7) as inputs.

                 //pull ups enabled in port D and A

                reset:
                   /* LDI    R16, low(RAMEND)
                   OUT    SPL, R16
                    LDI    R16, high(RAMEND)
                    OUT    SPH, R16               ;this code is for rcall, from ese 123
                   */
000000 e000        ldi r16, $00               ;load r16 with 0's
000001 bb01        out DDRD, r16              ;set up dip switch, port d as input
000002 bb04        out DDRC, r16
000003 ef0f        ldi r16, $ff               ;load r16 with 1's
000004 bb02        out PORTD, r16             ;set up the pull ups in D
000005 bb05        out PORTC, r16             ;set up the pull ups in C
000006 ef1f        ldi r17, $ff               ;delay timer    ///changed from 2 to ff
000007 e000        ldi r16, $00               ;load r16 with 1's
000008 bb08        out PORTB, r16             ;load PORT B, active low/turn on light
                //wait for the PBSW1 signal, activate or deactivate 7seg for every press
                //eliminate the debounces

                main_loop:

000009 9998        SBIC PINC, 0               ;wait for the button press
00000a cffe        rjmp main_loop             ;if button is not pressed repeat the loop
00000b e120        ldi r18, $10               ;reset delay timer   ///changed from 2 to 10
00000c e130        ldi r19, $10               ;reset second delay timer    ///changed from 2 to 10
00000d c000        rjmp delay1                ;delay for 10 clock cycles

                   //This will delay for 10000 us, which is equal to 10 ms

                delay1:
```

```
00000e 951a         dec r17                     ;decrement for 255 cycles
00000f f7f1         brne delay1                 ;loop for 255 cyles
000010 952a         dec r18                     ;decrement for 40 cycles
000011 ef1f         ldi r17,$ff                 ;set the ldi back to 255
000012 f7d9         brne delay1                 ;go back to delay1 if not 0
000013 c000         rjmp seg_on_off



                //turn on the light when the button pressed and when the button is 0
                //if its still pressed it will just keep the lit on, until 0 and
                // it will start the program over.

                seg_on_off:

000014 9998         SBIC PINC,0                 ;Check if the button is still pressed
000015 cff3         rjmp main_loop              ;repeat the program it is a noise
000016 9500         com r16                     ;invert all bits in r16
000017 bb08         out PORTB, r16              ;Turn on led

                check_button:
000018 9998         SBIC PINC,0                 ;check if the button is not pressed
000019 c003         rjmp delay2                 ;delay for 10ms for debounce
00001a 9998         SBIC PINC,0                 ;check if the button is still not pressed
00001b cfed         rjmp main_loop              ;restart the program
00001c cffb         rjmp check_button           ;loop again

                    //This will delay for 10000 us, which is equal to 10 ms.
                delay2:
00001d 951a         dec r17                     ;decrement for 255 cycles
00001e f7f1         brne delay2                 ;loop for 255 cyles
00001f 953a         dec r19                     ;decrement for 40 cycles
000020 ef1f         ldi r17,$ff                 ;set the ldi back to 255
000021 f7d9         brne delay2                 ;go back to delay1 if not 0
000022 cfe6         rjmp main_loop              ;restart the program

                /*

                //this is a subroutine
                delay:
                   dec r17                      ;decrement for 8 cycles
                   brne delay                   ;loop for 8 cyles
                   ldi r17, $2                  ;reset delay timer
                   ret                          ;return



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:  10 r17:   5 r18:   2 r19:   2 r20:   0 r21:   0 r22:   0 r23:   0
```

```
r24:   0 r25:   0 r26:   0 r27:   0 r28:   0 r29:   0 r30:   0 r31:   0
x  :   0 y  :   0 z  :   0
Registers used: 4 out of 35 (11.4%)

"ATmega16" 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  :   4 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   :   4 eor   :   0 fmul  :   0 fmuls :   0
fmulsu:   0 icall :   0 ijmp  :   0 in    :   0 inc   :   0 jmp   :   0
ld    :   0 ldd   :   0 ldi   :   8 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   :   6 pop   :   0
push  :   0 rcall :   0 ret   :   0 reti  :   0 rjmp  :   8 rol   :   0
ror   :   0 sbc   :   0 sbci  :   0 sbi   :   0 sbic  :   4 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: 7 out of 113 (6.2%)

"ATmega16" memory use summary [bytes]:
Segment   Begin    End      Code   Data   Used    Size   Use%
----------------------------------------------------------------
[.cseg] 0x000000 0x000046     70      0     70   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
```

```
AVRASM ver. 2.1.52  C:\Users\radra_000\Box Sync\college sophomore fall 2014\fall 2014 notes and files\ese 380 lab\lab 4\incr_decr\incr_decr\incr_decr.asm Tue Sep 30 20:54:13 2014

C:\Users\radra_000\Box Sync\college sophomore fall 2014\fall 2014 notes and files\ese 380 lab\lab 4\incr_decr\incr_decr\incr_decr.asm(18): Including file 'C:\Program Files (x86)\Atmel\Atmel Toolchain\AVR Assembler\Native\2.1.39.1005\avrassembler\Include\m16def.inc'


                 * incr_decr.asm
                 *
                 *  Created: 9/29/2014 5:02:01 PM
                 *   Author: Raymond
                 *
                 * Description: This program is used as a counter program for the seven segment led display
                 *     When pin 0 is pressed then the counter is incremented and the LED will display the current count
                 *     ranging from 0 to 9. When it goes over 9 then the overflow LED will be lit. When pin6 is pressed
                 *     then the counter will decrement. When pin7 is pressed then the program will display a 0 and the
                 *     program is reset. The program checks for the switch bounces.
                 // inputs : port c0, c6, c7, port d
                 // outputs: port b, port a0
                 *
                 */

                 .list

                  //load all the pins in port B, Port D and port A 0, as outputs, Port C (0,6,7) as inputs.
                 //pull ups enabled in port D and A
                 reset:
                    /*LDI R16, low(RAMEND)
                   OUT SPL, R16
                    LDI R16, high(RAMEND)
                    OUT SPH, R16               ;this code is for rcall, from ese 123
                   */
000000 ef0f         ldi r16, $ff               ;load r16 with 1's
000001 bb08         out PORTB, r16             ;set up the 7seg disp
000002 bb0a         out DDRA, r16
000003 e000         ldi r16, $00               ;load r16 with 0's
000004 bb04         out DDRC, r16
000005 bb01         out DDRD, r16              ;set up dip switch, port d as input
000006 ef0f         ldi r16, $ff               ;load r16 with 1's
000007 bb02         out PORTD, r16             ;set up the pull ups in D
000008 bb05         out PORTC, r16             ;set up the pull ups in C
000009 bb04         out DDRC, r16
00000a ec01         ldi r16, $c1               ;set up pullups for
00000b bb02         out PORTD, r16             ;the 8th 7th and 1st bit
00000c ef1f         ldi r17, $ff               ;delay timer
                    //ldi r19, $9              ;clear r19 for use compare for overflow
00000d e020         ldi r18, $00               ;clear r18 for use on counter
00000e e040         ldi r20, $00               ;clear for overflow
00000f 9ad8         SBI PORTA, 0               ;set the led to 1 //active low
                 main_loop:
000010 e170         ldi r23, $10               ;set the first delay counter to  10
000011 e180         ldi r24, $10               ;set the second delay coutner to 10
000012 e190         ldi r25, $10               ;set the third delay counter to  10
000013 e150         ldi r21, $10               ;set the fourth delay coutner to 10
000014 e160         ldi r22, $10               ;set the fifth delay coutner to 10
000015 9998         sbic PINC,0                ;check for button press increment
000016 c00b         rjmp dec_check             ;keep checking

                 delay1:
```

```
000017 951a         dec r17                     ;decrement for 255 cycles
000018 f7f1         brne delay1                 ;loop for 255 cyles
000019 957a         dec r23                     ;decrement for 40 cycles
00001a ef1f         ldi r17,$ff                 ;set the ldi back to 255
00001b f7d9         brne delay1                 ;go back to delay1 if not 0


00001c 9998         sbic PINC,0                 ;check for button press increment
00001d c004         rjmp dec_check              ;if false then check decrement button
00001e 3029         CPI r18,$09                 ;check if counter is at 9
00001f f0c9         breq overflow               ;if so then set overflow for led
000020 9523         inc r18                     ;inc counter for LED
000021 c018         rjmp bcd_7seg               ;jump to bcd display

                  dec_check:                    ;check for dec pushbutton
000022 999e         sbic PINC,6                 ;check if button is pressed
000023 c009         rjmp reset_check            ;if not jump to reset check
                    //rcall delay                   ;delay for bounce

                  delay2:

000024 951a         dec r17                     ;decrement for 255 cycles
000025 f789         brne delay1                 ;loop for 255 cyles
000026 958a         dec r24                     ;decrement for 40 cycles
000027 ef1f         ldi r17,$ff                 ;set the ldi back to 255
000028 f7d9         brne delay2                 ;go back to delay1 if not 0


000029 999e         sbic PINC, 6                ;check if button is pressed
00002a c002         rjmp reset_check            ;if not then check reset button
00002b 952a         dec r18                     ;if so then decrement
00002c c00d         rjmp bcd_7seg               ;display 7 seg leds

                  reset_check:
00002d 999f         sbic PINC,7                 ;check if reset button is pressed
00002e cfe1         rjmp main_loop              ;if not then go back to main loop
                    //rcall delay                   ;if so then delay for bounce

                  delay3:
00002f 951a         dec r17                     ;decrement for 255 cycles
000030 f731         brne delay1                 ;loop for 255 cyles
000031 959a         dec r25                 ;decrement for 40 cycles
000032 ef1f         ldi r17,$ff                 ;set the ldi back to 255
000033 f7d9         brne delay3                 ;go back to delay1 if not 0

000034 999f         sbic PINC,7                 ;and check if button is pressed
000035 cfda         rjmp main_loop              ;if not then check other buttons
000036 e420         ldi r18,$40                 ;if so then
000037 bb28         out PORTB, r18              ;display 0
000038 cfc7         rjmp reset                  ;and reset the program

                  overflow:
000039 98d8         cbi PORTA, 0;               ;turn on for overflow display

                  bcd_7seg:
00003a e0f0         ldi ZH, high (table*2)      ;set Z to start of table
00003b e0e0         ldi ZL, high (table*2)      ;
00003c e000         ldi r16, $00                ;clear for later use
00003d 0fe2         add ZL, r18                 ;add low byte
00003e 1ff0         adc ZH, r16                 ;add in the CY
00003f 9124         lpm r18,Z                   ;Load in the byte pattern from table into r18

                  display:
000040 bb4b         out PORTA, r20              ;ouput overflow
000041 bb28         out PORTB, r18              ;output the LEDs
000042 9b98         sbis PINC, 0                ;Check PINC0 for
```

```
000043 cffc         rjmp display                ;when it is not pressed
                    //rcall delay                   ;check for the

                  delay4:
000044 951a         dec r17                     ;decrement for 255 cycles
000045 f689         brne delay1                 ;loop for 255 cyles
000046 955a         dec r21                     ;decrement for 40 cycles
000047 ef1f         ldi r17,$ff                 ;set the ldi back to 255
000048 f7d9         brne delay4                 ;go back to delay1 if not 0



000049 9b98         sbis PINC, 0                ;debounce
00004a cff5         rjmp display                ;if debounce then check again
00004b 9b9e         sbis PINC, 6                ;Check PINC6 for
00004c cff3         rjmp display                ;when it is not pressed
                    //rcall delay                   ;check for the

                  delay5:
00004d 951a         dec r17                     ;decrement for 255 cycles
00004e f641         brne delay1                 ;loop for 255 cyles
00004f 956a         dec r22                     ;decrement for 40 cycles
000050 ef1f         ldi r17,$ff                 ;set the ldi back to 255
000051 f791         brne delay4                 ;go back to delay1 if not 0

000052 9b9e         sbis PINC, 6                ;debounce
000053 cfec         rjmp display                ;if debounce then check again
000054 cfbb         rjmp main_loop
                    ;table of seven segment display 0-9
000055 7940
000056 3024
000057 1219
000058 7803
000059 1800       table: .db $40,$79,$24,$30,$19,$12,$03,$78,$0,$18
                    ;       0   1   2   3   4   5   6   7   8   9
                  /*//this is a subroutine
                  delay:
                    dec r17                     ;decrement for 8 cycles
                    brne delay                  ;loop for 8 cyles
                    ldi r17, $2                 ;reset delay timer
                    ret                         ;return


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:  14 r17:  11 r18:   9 r19:   0 r20:   2 r21:   2 r22:   2 r23:   2
r24:   2 r25:   2 r26:   0 r27:   0 r28:   0 r29:   0 r30:   2 r31:   2
x  :   0 y  :   0 z  :   1
Registers used: 12 out of 35 (34.3%)

"ATmega16" instruction use summary:
```

```
.lds  :   0 .sts  :   0 adc   :   1 add   :   1 adiw  :   0 and   :   0
andi  :   0 asr   :   0 bclr  :   0 bld   :   0 brbc  :   0 brbs  :   0
brcc  :   0 brcs  :   0 break :   0 breq  :   1 brge  :   0 brhc  :   0
brhs  :   0 brid  :   0 brie  :   0 brlo  :   0 brlt  :   0 brmi  :   0
brne  :  10 brpl  :   0 brsh  :   0 brtc  :   0 brts  :   0 brvc  :   0
brvs  :   0 bset  :   0 bst   :   0 call  :   0 cbi   :   1 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   :   1 cpse  :   0 dec   :  11 eor   :   0 fmul  :   0 fmuls :   0
fmulsu:   0 icall :   0 ijmp  :   0 in    :   0 inc   :   1 jmp   :   0
ld    :   0 ldd   :   0 ldi   :  21 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   :  11 pop   :   0
push  :   0 rcall :   0 ret   :   0 reti  :   0 rjmp  :  14 rol   :   0
ror   :   0 sbc   :   0 sbci  :   0 sbi   :   1 sbic  :   6 sbis  :   4
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: 15 out of 113 (13.3%)

"ATmega16" memory use summary [bytes]:
Segment   Begin    End      Code   Data   Used    Size   Use%
----------------------------------------------------------------
[.cseg] 0x000000 0x0000b4    170     10    180   16384   1.1%
[.dseg] 0x000060 0x000060      0      0      0    1024   0.0%
[.eseg] 0x000000 0x000000      0      0      0     512   0.0%

Assembly complete, 0 errors, 0 warnings
```