EE 354 **Sept. 2, 2016**
A51 Sample Programs

Note: These programs are meant to be run on the simulator and are used to illustrate characteristics of the 8051 assembly code.

```
;AsmDoubleAdd
;Adds memory locations 40-41 to memory 42-43
;  puts result in 43-44
; Use the simulator with View - memory2
;  In memory window look at i:40 and preload values in
;  40-42
MainSeg SEGMENT CODE
;
CSEG at 0h
   ljmp Start
RSEG MainSeg
Start: mov r0, #40
       mov a, @r0
       inc r0
       inc r0
       add a, @r0
       inc r0
      inc r0
      mov @r0, a
      mov r0, #41
      mov a, @r0
      inc r0
      inc r0
      addc a, @r0
      inc r0
      inc r0
      mov @r0, a
end
```

```
;AsmMacro
;This program illustrates the use of simple macros.  A Delay macro
;  is created with a parameter.
;
MyCode SEGMENT CODE
;
Delay MACRO x
LOCAL Lp
      mov R5, #x
  Lp: djnz R5, Lp
  ENDM
;
CSEG at 0000h                  ;Reset vector
     LJMP Start
;User code for main program
       RSEG    MyCode
;
Start:
       mov R1, #1
 Top:  mov P0, R1
       Delay 5
       mov P0, R2
       Delay 10
       inc R1
       dec R2
       sjmp Top
 END
```

```
;Bin2Dec
;This program converts a binary number in a to decimal and outputs the
;  decimal equivalent to P0.
; ********************************************************************
MyCode SEGMENT CODE
;
;
CSEG at 0000h                   ;Reset vector
      LJMP Start
;User code for main program
       RSEG    MyCode
;
Start:
       mov sp, #7       ;Set sp above RB0
  Lp:  mov a, #41H      ;Data in a
       call Convert
       mov P0, a        ;output answer to P0
       sjmp Lp          ;Loop forever

Convert: mov b, #10       ;10 decimal into B
         div ab           ;divide a by b. a has answer, b had remainder
         swap a           ;move answer in a to high nibble
         orl a, b         ;mov remainder to low nibble
         ret
        END
```

```
;*********************************************************************
;Mpyx10
;  Multiplies a 16 bit number in r3 r2 by 10.  Result goes to r5 r4
;  Does multiplication by shifting since 10 = 8 + 2.
;*********************************************************************
MyCode SEGMENT CODE
;
CSEG at 0000h                   ;Reset vector
      LJMP Start
;User code for main program
       RSEG    MyCode
;
Start:
       mov r2, #0255  ;255 base 10 in r3, r2
       mov r3, #0
       mov a, r2      ;get least byte
       call shift16
       mov a, r2      ;make r5,r4 = r3,r2 * 2
       mov r4, a
       mov a, r3
       mov r5, a
       call shift16
       call shift16
       mov a, r2      ;r3,r2 = r3,r2 * 8
       add a, r4      ;add r3,r2 and r5,r4
       mov r2, a
       mov a, r3
       addc a, r5
       mov r3, a
 loop: sjmp loop

 shift16:clr c      ;16 bit shift of r3,r2
         mov a, r2
         rlc a
         mov r2, a
         mov a, r3
         rlc a
         mov r3, a
         ret
    END
```

```
;RBank
;This program illustrates the use of bank switching.
;Run this program in the simulator and watch internal memory beginning
;  at location 0 (memory window to I:0) and register R5
MainSeg SEGMENT CODE
DelaySeg SEGMENT CODE
;
CSEG at 0h
  ljmp Start
RSEG MainSeg
Start: mov SP, #16       ;SP above RB1 for return address
Top:   mov a, #55h
       mov P1, a
       mov R5, a         ;This added to show that R5 is protected
       call ShortDelay
       mov a, #0aah
       mov P1, a
       mov R5, a
       call ShortDelay
       sjmp Top

;Delay
RSEG DelaySeg
ShortDelay:
       setb rs0
       clr rs1
       mov R5, #10h     ;R5 in Register Bank 1
Here:  djnz R5, Here
       clr rs0
       ret
END
```

```
;*********************************************************************
;AsmBankSW
;This program loads R2 and R3 with data and calls two subprograms which
;  shift the data one place to the left.  The first subprogram saves
;  the PSW on the stack and uses the current register bank.  The second
;  subprogram saves the PSW on the stack and switches register banks to
;  RB1.
; *********************************************************************
MyCode SEGMENT CODE
Shift  SEGMENT CODE
;
CSEG at 0000h                   ;Reset vector
      LJMP Start
;User code for main program
       RSEG    MyCode
;
Start:
       mov sp, #15      ;Set sp above RB2
       setb c           ;Set carry just for fun
       mov a, #89h      ;Dat in a just for fun
       mov r2, #0AAH
       mov r3, #0FH
   LP: call ShiftLeft
       mov 10, r2
       mov 11, r3
       call ShiftLeftRB
       mov r2, 10
       mov r3, 11
       sjmp LP

       RSEG Shift
ShiftLeft:push PSW     ;This version pushes the psw and restores it
          mov a, r2
          rlc a
          mov r2, a
          mov a, r3
          rlc a
          mov r3, a
          pop PSW
          ret
ShiftLeftRB:push PSW     ;This version switches register banks
           setb RS0
           clr  RS1
           mov a, r2
           rlc a
           mov r2, a
```

```
        mov a, r3
        rlc a
        mov r3, a
        clr RS0
        pop PSW
        ret
END
```

```
;*********************************************************************
;BCDAdd
;  Adds two 16 bit numbers using BCD arithmetic
;  r1 r0 + r3 r2 -> r5 r4
;*********************************************************************
MyCode SEGMENT CODE
;
CSEG at 0000h                   ;Reset vector
      LJMP Start
;User code for main program
       RSEG    MyCode
;
Start:
       mov r1, #01H      ;R1 R0 = 0145
       mov r0, #45H
       mov r3, #02H      ;R3 R2 = 0239
       mov r2, #39H
       mov a, r0
       add a, r2
       da a
       mov r4, a
       mov a, r1
       addc a, r3
       da a
       mov r5, a
  LP:  sjmp LP
  END
```

Note that this program assumes an 8051 variant with an onboard A to D converter and an external D to A converter on P2.

```
;*********************************************************************
;A2D2AAsm
;  Inputs from Channel 0 on the A to D Converter and outputs
;   8 most significant bits to P2 for the D to A converter.
;*********************************************************************
MyCode SEGMENT CODE
;
/* AT89C51AC3 EQU */
CKCON  EQU 08FH;       // Clock Control
ADCLK  EQU 0F2h;       // ADC Clock Control Register
ADCON  EQU 0F3h;       // ADC Control Register
ADDL   EQU 0F4h;       // ADC Data Low Byte
ADDH   EQU 0F5h;       // ADC Data High Byte
ADCF   EQU 0F6h;       // ADC Config Register
P4   EQU 0C0h
CSEG at 0000h                  ;Reset vector
       LJMP Start
;User code for main program
       RSEG    MyCode
;
Start: mov ADCF, #01h                  ;P1.0 = ADC[0]
       mov ADCON, #20h                 ;Enable ADC Function
       mov ADCLK, #0h                  ;AD Clock = Crystal/64
 Lp:   anl ADCON, #0F8h                ;Reset AtoD channel select
         orl ADCON, #0h                ;Select channel 0 on AN0
         orl ADCON, #020h              ;Enable ADC
         orl ADCON, #08h               ;Start conversion
 Lp1:    mov a, ADCON                  ;Wait for conversion to finish
           anl a, #10h
         jz Lp1
         mov P2, ADDH                  ;Send most significant byte to P2
         clr P4.0                      ;Toggle pin 4.0 (write to D to A)
         setb P4.0                     ;Clear ADC done bit
         anl ADCON, #0EFh              ;Loop forever
       sjmp Lp
END
```

```
;*********************************************************************
;ASMPWM
;Puts a PWM ramp function on PWM0 which comes out on P1.3
;  R3 holds the data.  This register is incremented about every tenth
;  of a second.  R3 is copied into CCAP0H and is used to determine
;  the value of the duty cycle.  PWM automatically loads CCAP0H into
;  CCAP0L and counts up on the PCA timer.  If the PCA timer is less than
;  the value in CCAP0L, P1.3 is low.  When the PCA timer is greater than
;  the value in CCAP0L, P1.3 is high.  Every time the PCA timer
overflows
;  CCAP0L is reloaded from CCAP0H.
;*********************************************************************
MyCode SEGMENT CODE
;
/* AT89C51AC3 EQU */
CCON    EQU 0D8h;  PCA Timer/Counter Control
CMOD    EQU 0D9h;  PCA Timer/Counter Mode
CL      EQU 0E9h;  PCA Timer/Counter Low byte
CH      EQU 0F9h;  PCA Timer/Counter High byte
CCAPM0     EQU 0DAh;  PCA Timer/Counter Mode 0
CCAPM1     EQU 0DBh;  PCA Timer/Counter Mode 1
CCAPM2     EQU 0DCh;  PCA Timer/Counter Mode 2
CCAPM3     EQU 0DDh;  PCA Timer/Counter Mode 3
CCAPM4     EQU 0DEh;  PCA Timer/Counter Mode 4
CCAP0H     EQU 0FAh;  PCA Compare Capture Module 0 H
CCAP1H     EQU 0FBh;  PCA Compare Capture Module 1 H
CCAP2H     EQU 0FCh;  PCA Compare Capture Module 2 H
CCAP3H     EQU 0FDh;  PCA Compare Capture Module 3 H
CCAP4H     EQU 0FEh;  PCA Compare Capture Module 4 H
CCAP0L     EQU 0EAh;  PCA Compare Capture Module 0 L
CCAP1L     EQU 0EBh;  PCA Compare Capture Module 1 L
CCAP2L     EQU 0ECh;  PCA Compare Capture Module 2 L
CCAP3L     EQU 0EDh;  PCA Compare Capture Module 3 L
CCAP4L     EQU 0EEh;  PCA Compare Capture Module 4 L
P4      EQU 0C0h;  Port 4
CSEG at 0000h                   ;Reset vector
      LJMP Start
;User code for main program
       RSEG    MyCode
;
Start: mov CMOD, #02h           ;Clock for PCA is FPca/2
       mov CCON, #40h           ;Turn on PCA timer
       mov CCAPM0, #42h;        ;Enable PWM and Compare mode for PWM0
       mov R3, #0
   LP: mov CCAP0H, R3           ;Put output data in PWM register
       inc R3                   ;Increment data
```

```
       call Wait
       sjmp LP

Wait:  mov R5, #02h            ;This delay loop takes about a tenth
  LP3: mov R7, #0h             ;  of a second
  LP1: mov R6, #0h
  LP2:   djnz R6, LP2
       djnz R7, LP1
       djnz R5, LP3
       ret

END
```