**AVR Assembler Programming Tutorial**

**(1)**

**/\* Write a program to find the sum of 8 bit variables A and B.**

**\* For this programming problem the sum may be greater than 255 if A and B**

**\* are unsigned or less than -128 and greater than 127 if signed.**

**\* Store the sum into 16 bit variable C using little endian byte ordering.**

**\* C = A + B**

**\*/**

.INCLUDE <m328pdef.inc>

.DSEG

A: .BYTE 1

B: .BYTE 1

C: .BYTE 2

.CSEG

Adder816:

; load

clr r1 ; r1:r0 = 0:A

lds r0,A

clr r3 ; r3:r2 = 0:B

lds r2,B

; add

add r0,r2 ; add least significant bytes

adc r1,r3 ; add with carry most significant bytes

; store

sts C,r0 ; store least significant byte first

sts C+1,r1

rjmp Adder816

**(2) Adding 16-bit Numbers adding two 16-bit numbers and store in given RAM Address 0x85 (in form of little endian). The code for this is shown below.**

.include "m32def.inc"

.def num1L = r16 ; define lower byte of number 1 as r16

.def num1H = r17 ; define upper byte of number 1 as r17

.def num2L = r18 ; define lower byte of number 2 as r18

.def num2H = r19 ; define upper byte of number 2 as r19

.EQU RAM\_ADDR = 0x80

.cseg

.org 0x00

ldi num1L,0x34 ; load 0x34 into r16

ldi num1H,0x12 ; load 0x12 into r17

ldi num2L,0xCD ; load 0xCD into r18

ldi num2H,0xAB ; load 0xAB into r19

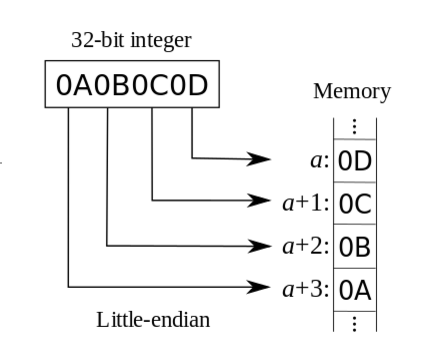
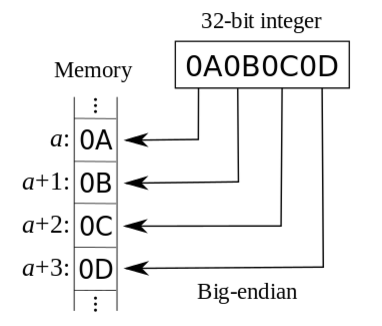
add num1L,num2L ; add lower bytes of number

adc num1H,num2H ; add upper bytes of number

sts 0x85, num1L

sts 0x85, num1H

loop: rjmp loop ; infinite loop

**(3)**

**/\* Given variables A and B, each holding an 8-bit signed 2's complement number,**

**\* write a program to find the maximum value and put into variable C. For**

**\* example if A > B then C = A.**

\*

\* Solution B: Basic implementation of if-then-else statement.

\* Structure modified to immediately store result.

\*/

.INCLUDE <m32def.inc>

.DSEG

A: .BYTE 1

B: .BYTE 1

C: .BYTE 1

.CSEG

Max2:

lds r16,A ; load

lds r17,B

cp r16,r17

brlt elseMax2 ; if (A >= B)

sts C,r16 ; then C = A

rjmp endMax2

elseMax2:

sts C, r17

endMax2:

**(4)Finding a given number in an Array store in Flash Rom and output to PORTC and PORTB**

.include "m32def.inc"

.ORG $0

.MACRO IO\_CONFIG

LDI R20,@1

OUT @0,R20

.ENDMACRO

IO\_CONFIG DDRB,0XFF

IO\_CONFIG DDRC,0XFF

LDI ZH,HIGH(TABLE<<1)

LDI ZL,LOW(TABLE<<1)

LDI R16,0

LDI R18,9

LDI R17,0

LOOP:

LPM R17,Z+

CPI R17,'.'

BREQ HERE

CP R16,R18

BREQ RUN

INC R16

RJMP LOOP

RUN:

OUT PORTC,R17

OUT PORTB,R17

HERE: RJMP HERE

.ORG $500

TABLE: .DB 0,1,2,3,4,5,6,7,8,9,'.'

**(5) CONVERTING 4 PACKED BCD NUMERS IN FLASH ROM TO ASCII CODE AND STORE TO RAM ADDRESS $80**

.INCLUDE "M32DEF.INC"

.EQU RAM\_ADDR = 0x80

LDI R16,HIGH(RAMEND)

OUT SPH,R16

LDI R16,LOW(RAMEND)

OUT SPL,R16 ;SP = RAMEND

RCALL BCD\_ASCII\_COV

HERE: RJMP HERE

;-----convert packed BCD to ASCII

BCD\_ASCII\_COV:

LDI ZL,LOW(MYBYTE<<1)

LDI ZH,HIGH(MYBYTE<<1) ; Z = MYBYTE

LDI XL,LOW(RAM\_ADDR)

LDI XH,HIGH(RAM\_ADDR) ;X = RAM\_ADDR

LDI R16,4 ; COUNTER = 4

L1:

LPM R20,Z+

MOV R21,R20 ;R21 = R20

ANDI R21,0x0F ;mask the upper nibble

ORI R21,0x30 ;make it an ASCII

ST X+,R21

SWAP R20 ;swap the nibbles

ANDI R20,0x0F ;mask the upper nibble

ORI R20,0x30 ;make it an ASCII

ST X+,R20

DEC R16 ;decrement COUNTER

BRNE L1 ;loop while COUNTER is not zero

RET

;-----send ASCII to Port B

SEND\_TO\_PORTB:

LDI XH,HIGH(RAM\_ADDR)

LDI XL,LOW(RAM ADDR) ;X = RAM ADDR

LDI R16,8 ;COUNTER = 8

L2: LD R20,X+

OUT PORTB, R20 ; PORTB = R20

DEC R16 ;decrement counter

BRNE L2 ;loop while counter

RET

MYBYTE: .DB 0x25, 0x67, 0x39, 0x52

**(6) CHECKSUM**

.INCLUDE "M32DEF.INC"

.EQU OPTION\_SIZE = 0x4

.EQU RAM\_OPTIONS = 0x100

;-------------main program

.ORG 0

LDI R16,HIGH(RAMEND)

OUT SPH,R16

LDI R16,LOW(RAMEND)

OUT SPL,R16 ;SP points to RAMEND

RCALL LOAD\_OPTIONS ;load options

RCALL TEST\_CHKSUM ;test checksum

TST R20

BREQ L1 ;if data is not corrupted go to L1

RCALL INIT\_OPTIONS ;initialize options

L1: ;Here you can use the options

RCALL CAL\_CHKSUM ;calculating checksum

RCALL STORE\_OPTIONS ;storing options in EEPROM

HERE: RJMP HERE

;-----Load R20 with contents of location X of EEPROM

LOAD\_FROM\_EEPROM:

SBIC EECR, EEWE

RJMP LOAD\_FROM\_EEPROM ;wait while EEPROM is busy

OUT EEARH,XH

OUT EEARL,XL ;EEAR = X

SBI EECR,EERE ;set Read Enable to one

IN R20,EEDR ;load EEPROM Data Register to R20

RET

;-----Store R20 into location X of EEPROM

STORE\_IN\_EEPROM:

SBIC EECR, EEWE

RJMP STORE\_IN\_EEPROM ;wait while EEPROM is busy

OUT EEARH,XH

OUT EEARL,XL ;EEAR = X

OUT EEDR,R20

SBI EECR,EEMWE ;set Master Write Enable to one

SBI EECR,EEWE ;write EEDR into EEPROM

RET

;-----copying the data from EEPROM to internal SRAM

LOAD\_OPTIONS:

LDI XL,LOW(E\_OPTIONS)

LDI XH,HIGH(E\_OPTIONS) ;X points to E OPTIONS

LDI YL,LOW(RAM\_OPTIONS)

LDI YH,HIGH(RAM\_OPTIONS) ;Y points to RAM OPTIONS

LDI R16,OPTION\_SIZE+1 ;COUNTER = OPTION SIZE + 1

LL1: CALL LOAD\_FROM\_EEPROM ;load R20 with EEPROM loc X

ST Y+,R20 ;store R20 in RAM loc Y

INC XL ;increment XL

BRNE LL2 ;if not carry go to LL2

INC XH

LL2: DEC R16 ;decrement COUNTER

BRNE LL1 ;if COUNTER not zero go to LL1

RET

;-----copying the data from internal SRAM to EEPROM

STORE\_OPTIONS:

LDI XL,LOW(E\_OPTIONS)

LDI XH,HIGH(E\_OPTIONS) ;X points to E OPTIONS

LDI YL,LOW(RAM\_OPTIONS)

LDI YH,HIGH(RAM\_OPTIONS) ;Y points to RAM\_OPTIONS

LDI R16,OPTION\_SIZE+1 ;COUNTER= OPTION\_SIZE+ 1

SL1: LD R20, Y+

CALL STORE\_IN\_EEPROM ;store R20 in loc X

INC XL ;increment XL

BRNE SL2 ;if not carry go to SL2

INC XH

SL2: DEC R16 ;decrement COUNTER

BRNE SL1 ;loop while COUNTER is not zero

RET

; Init

INIT\_OPTIONS:

LDI ZL,LOW(FLASH\_OPTIONS<<1)

LDI ZH,HIGH(FLASH\_OPTIONS<<1) ;Z points to FLASH OPTIONS

LDI YL,LOW(RAM\_OPTIONS)

LDI YH,HIGH(RAM\_OPTIONS) ;Y points to RAM\_OPTIONS

LDI R16,OPTION\_SIZE ;COUNTER = OPTION SIZE

H1: LPM R18,Z+ ;load R18 with program mem. location Z

ST Y+,R18 ;store R18 in loc Y of RAM

DEC R16 ;decrement COUNTER

BRNE H1 ;if COUNTER is not zero go to H1

RET

;-------calculating checksum byte

CAL\_CHKSUM:

LDI YL,LOW(RAM\_OPTIONS)

LDI YH,HIGH(RAM\_OPTIONS) ;Y points to RAM OPTION

LDI R16,OPTION\_SIZE ;COUNTER = OPTION SIZE

LDI R20,0 ;SUM = 0

CL1: LD R17,Y+ ; load R17 with contents of loc y

ADD R20,R17 ;SUM = SUM + R17

DEC R16 ;decrement COUNTER

BRNE CL1 ;if COUNTER is not zero go to CL1

NEG R20 ;two's complement SUM

ST Y,R20 ;store checksum in loc Y of RAM

RET

;-------testing checksum byte

TEST\_CHKSUM:

LDI YL,LOW(RAM\_OPTIONS)

LDI YH,HIGH(RAM\_OPTIONS) ;Y points to RAM OPTIONS

LDI R16,OPTION\_SIZE+1

LDI R20,0 ;SUM = 0

TL1: LD R17,Y+ ;load R17 with contents of loc Y

ADD R20,R17 ;SUM = SUM + R17

DEC R16 ;decrement COUNTER

BRNE TL1 ;loop while COUNTER is not zero

RET

;-------initial values in program ROM

FLASH\_OPTIONS: .DB 0x25,0x62,0x3F,0x52

;-------EEPROM

.ESEG

.ORG $0

E\_OPTIONS: .DB 0x25,0x62,0x3F,0x52

**(7) CONVERTING BINARY TO ASCII**

.INCLUDE "M32DEF.INC"

.DEF NUM = R20

.DEF DENOMINATOR = R21

.DEF QUOTIENT = R22

.EQU RAM\_ADDR = 0x200

.EQU ASCII\_RESULT = 0x210

;--------------main program

.ORG 0

LDI R18,HIGH(RAMEND)

OUT SPH,R18

LDI R18,LOW(RAMEND)

OUT SPL,R18

LDI R16,0x00

OUT DDRA,R16 ; PORTA is Input Port

RCALL BIN\_DEC\_CONVRT

RCALL DEC\_ASCI\_CONVRT

END: RJMP END

;-----------Converting BIN(HEX) TO DEC (00-FF TO 000-255)

BIN\_DEC\_CONVRT:

LDI XL,LOW(RAM\_ADDR) ;save DEC digits in these locations

LDI XH,HIGH(RAM\_ADDR)

IN NUM,PINA ;read data from PORT A

LDI DENOMINATOR,10

RCALL DIVIDE ;QUOTIENT=PINA/10, NUM= PINA%10

ST X+,NUM ;save lower digit

MOV NUM,QUOTIENT

RCALL DIVIDE ;divide by 10 once more

ST X+,NUM ;save the next digit

ST X+,QUOTIENT ;save the last digit

RET

; BCD to ASCII Conversion

DEC\_ASCI\_CONVRT:

LDI XL,LOW(RAM\_ADDR) ; addr. of DEC data

LDI XH,HIGH(RAM\_ADDR)

LDI YL,LOW(ASCII\_RESULT) ;addr. of ASCII data

LDI YH,HIGH(ASCII\_RESULT)

LDI R16,3 ; count

BACK: LD R20,X+ ;get DEC digit

ORI R20,0x30 ;make it an ASCII digit

ST Y+,R20 ;store it

DEC R16 ;decrement counter

BRNE BACK ;repeat until the last one

RET

; 8-bit integer division

DIVIDE:

LDI QUOTIENT,0

L1: INC QUOTIENT

SUB NUM, DENOMINATOR

BRCC L1

DEC QUOTIENT

ADD NUM, DENOMINATOR

RET

**(8) CONVERTING BINARY TO ASCII USING LOOP**

.INCLUDE "M32DEF.INC"

.DEF NUM = R20

.DEF DENOMINATOR = R21

.DEF QUOTIENT = R22

.EQU RAM\_ADDR = 0x200

.EQU ASCII\_RESULT = 0x210

;--------------main program

.ORG 0

LDI R18,HIGH(RAMEND)

OUT SPH,R18

LDI R18,LOW(RAMEND) ; SP = RAMEND

OUT SPL,R18

LDI R16,0x00

OUT DDRA,R16 ; PORTA is Input Port

RCALL BIN\_ASCI\_CONVRT

END: RJMP END

;-----------Converting BIN(HEX) TO ASCII

BIN\_ASCII\_CONVRT:

LDI XL,LOW(ASCII\_RESULT) ;save results in these loc.

LDI XH,HIGH(ASCII\_RESULT)

IN NUM,PINA ;read data from PORT A

LDI DENOMINATOR,10

RCALL DIVIDE ;QUOTIENT=PINA/10, NUM= PINA%10

ORI NUM, 0x30 ;make it an ASCII digit

ST X+,NUM ;save lower digit

MOV NUM,QUOTIENT

RCALL DIVIDE ;divide by 10 once more

ORI NUM, 0x30 ;make it an ASCII digit

ST X+,NUM ;save the next digit

ORI QUOTIENT, 0x30 ;make it an ASCII digit

ST X+,QUOTIENT ;save the last digit

RET

; 8-bit integer division

DIVIDE:

LDI QUOTIENT,0

L1: INC QUOTIENT

SUB NUM, DENOMINATOR

BRCC L1

DEC QUOTIENT

ADD NUM, DENOMINATOR

RET

**(9) toggling Port B using macros**

.INCLUDE "M32DEF.INC"

.MACRO LOADIO

LDI R20,@1

OUT @0,R20

.ENDMACRO

;--------------------------time delay macro

.MACRO DELAY

LDI @0,@1

BACK:

NOP

NOP

NOP

NOP

DEC @0

BRNE BACK

.ENDMACRO

;--------------------------program starts

.ORG 0

.LISTMAC

LOADIO DDRB,0xFF ;make PORTB output

L1: LOADIO PORTB,0x55 ; PORTB = 0x55

DELAY R18,0x70 ;delay

LOADIO PORTB,0xAA ; PORTB = 0xAA

DELAY R18,0x70 ;delay

RJMP L1

**(10)EEPROM PROGRAM TO WRITE AND READ**

LDI R16,0XFF

OUT DDRB,R16

LDI R20,0x55

WAIT\_WRITE:

SBIC EECR,EEWE

RJMP WAIT\_WRITE

LDI R18,0

LDI R17,0x5F

OUT EEARH,R18

OUT EEARL,R17

OUT EEDR,R20

SBI EECR,EEMWE

SBI EECR,EEWE

WAIT\_READ:

SBIC EECR,EEWE

RJMP WAIT\_READ

LDI R18,0

LDI R17,0x5F

OUT EEARH,R18

OUT EEARL,R17

SBI EECR,EERE

IN R16,EEDR

OUT PORTB,R16

HERE: JMP HERE