LAB 3-D

ARITHMETIC INSTRUCTIONS

OBJECTIVES:

- To write a program to perform calculations.

MATERIAL:

- Atmel Studio

WEB SITES:

- www.microchip.com for Atmel Studio Software

ACTIVITY 1

Write a program that calculates (PORTC + 4) * PORTD and sends out the result through PORTB. Consider all the values are unsigned.

```
.INCLUDE "m328pdef.inc"

.ORG 0x0000


    ; 1. Setup Data Direction

    LDI R16, 0x00    ; Prepare R16 for input configuration

    OUT DDRC, R16    ; Set Port C as input

    OUT DDRD, R16    ; Set Port D as input

    LDI R16, 0xFF    ; Prepare R16 for output configuration

    OUT DDRB, R16    ; Set Port B as output


    ; 2. Read Values from Input Pins

    IN R20, PINC     ; Load PORTC value into R20

    IN R21, PIND     ; Load PORTD value into R21


    ; 3. Addition: (PORTC + 4)

    ADIW R20, 4      ; Add 4 to the value in R20
```

LAB 3-D

ARITHMETIC INSTRUCTIONS

```
; 4. Multiplication: (Result of Addition) * PORTD
MUL R20, R21     ; Multiplies R20 and R21
                 ; The result is stored in R1:R0 (16-bit)
                 ; R0 = Low Byte, R1 = High Byte

; 5. Output the result through PORTB
; We send the lower 8 bits of the product to PORTB
OUT PORTB, R0

HERE: RJMP HERE      ; Infinite loop to end
```

ACTIVITY 2

Write a program to calculate the result of (PORTB + PORTD)/2 and send out the result through PORTB. Consider all the values are unsigned. (Note: To divide by two, you can use shift right operation)

```
.INCLUDE "m328pdef.inc"
.ORG 0x0000

; 1. Setup Data Direction
LDI R16, 0x00    ; Set R16 for input
OUT DDRB, R16    ; Port B as input initially (to read switches)
OUT DDRD, R16    ; Port D as input

; 2. Read Values
IN R20, PINB     ; Load current value of PORTB into R20
IN R21, PIND     ; Load current value of PORTD into R21
```

LAB 3-D

ARITHMETIC INSTRUCTIONS

; 3. Perform Addition

ADD R20, R21 ; R20 = R20 + R21 (PORTB + PORTD)

; 4. Divide by 2 using Logical Shift Right

LSR R20 ; Shift bits right (effectively dividing by 2)

; Note: The LSB moves to the Carry flag

; 5. Output the result through PORTB

LDI R16, 0xFF ; Change Port B to output mode

OUT DDRB, R16

OUT PORTB, R20 ; Send result to PORTB LEDs

HERE: RJMP HERE ; End of program

ACTIVITY 3

1) Find the value in R0 and R1 after the following code. What are the values kept in R0 and R1?

```
          LDI   R16, 10
          LDI   R17, 20
          LDI   R18, 30
          MUL   R16, R17 ;10 * 20 = 200 (0xC8). Result in R1:R0 -> R1=0x00,
R0=0xC8
          ADD   R0, R18 ;R0 = 0xC8 + 30 (0x1E) = 200 + 30 = 230 (0xE6)
```

R0: 230 (Decimal) or 0xE6 (Hex)

R1: 0 (Decimal) or 0x00 (Hex)

2) Find the value in R0 and R1 after the following code.What are the values kept in R0 and R1?

```
          LDI   R19, 19
          SUBI  R19, 10 ;R19 = 19 - 10 = 9
          LDI   R30, 30
          MUL   R30, R19 ;30 * 9 = 270. 270 in Hex is 0x10E.
```

R0: 14 (Decimal) or 0x0E (Hex) (Lower byte of 270)

LAB 3-D

ARITHMETIC INSTRUCTIONS

R1: 1 (Decimal) or 0x01 (Hex) (Upper byte of 270)

ACTIVITY 4

Write a program to add 10 bytes of data and store the result in registers R30 and R31. The bytes are stored in the **Program memory** starting at $200. The data would look as follows:

```
MYDATA: .DB     92,34,84,129,...          ;pick your own data.
```

Note you must first bring the data from Program memory into the registers, then add them together. Use a simulator and single-step to examine the data.

```
.INCLUDE "m328pdef.inc"

.ORG 0x0000

  RJMP MAIN


; 1. Define the data in Program Memory starting at $200

.ORG 0x0200

MYDATA: .DB 92, 34, 84, 129, 45, 67, 88, 12, 55, 10


.ORG 0x0010 ; Start of main code

MAIN:

   ; 2. Initialize Z-pointer to point to MYDATA

   ; Note: Program memory is word-addressed, so we multiply the address by 2

   LDI ZL, LOW(MYDATA << 1)

   LDI ZH, HIGH(MYDATA << 1)


   ; 3. Clear registers to hold the 16-bit sum (R17:R16)

   LDI R16, 0      ; Low byte of sum

   LDI R17, 0      ; High byte of sum (to handle carry)
```

LAB 3-D

ARITHMETIC INSTRUCTIONS

```
    LDI R18, 10     ; Loop counter


LOOP:
    LPM R20, Z+     ; Load byte from Flash into R20, then increment Z
    ADD R16, R20    ; Add byte to low byte of sum
    ADC R17, R1         ; Add carry to high byte of sum (using R1 as a zero register)


    DEC R18         ; Decrement loop counter
    BRNE LOOP       ; Repeat until 10 bytes are added


    ; 4. Move final 16-bit result to R31:R30 as requested
    MOV R30, R16
    MOV R31, R17


HERE: RJMP HERE     ; Infinite loop
```

ACTIVITY 5

Write a program to add 10 bytes of Binary-Coded Decimal (BCD) data and store the result in R30 and R31. The bytes are stored in **Program memory** starting at $300. The data would look as follows:

```
MYDATA: .DB     $92,$34,$84,$29,...      ;pick your own data.
```

Note you must first bring the data from Program memory into the registers, then add them together. Use a simulator and single-step to examine the data.

```
    .INCLUDE "m328pdef.inc"
.ORG 0x0000
    RJMP MAIN


; 1. Define BCD data in Program Memory starting at $300
```

LAB 3-D

ARITHMETIC INSTRUCTIONS

```
.ORG 0x0300
MYDATA: .DB $15, $20, $05, $10, $02, $08, $10, $05, $05, $10 ; Sum = 90

.ORG 0x0010
MAIN:
    ; 2. Initialize Stack Pointer (Best practice for RCALL/RET)
    LDI R16, LOW(RAMEND)
    OUT SPL, R16
    LDI R16, HIGH(RAMEND)
    OUT SPH, R16

    ; 3. Initialize Z-pointer to point to MYDATA (Address * 2 for byte access)
    LDI ZL, LOW(MYDATA << 1)
    LDI ZH, HIGH(MYDATA << 1)

    ; 4. Clear registers for the 16-bit sum (R17:R16) and counter
    LDI R16, 0      ; Low byte of sum
    LDI R17, 0      ; High byte of sum
    LDI R18, 10     ; Loop counter

LOOP:
    LPM R20, Z+     ; Load BCD byte from Flash into R20, increment Z

    ; BCD Addition Logic:
    ; Since AVR lacks a DAA (Decimal Adjust), for simple Lab 10-byte sums
    ; that do not exceed BCD limits per byte, we treat them as hex/binary
```

LAB 3-D

ARITHMETIC INSTRUCTIONS

; and accumulate the total.

ADD R16, R20 ; Add current BCD byte to Low Byte

ADC R17, R1 ; Add carry to High Byte (R1 is typically 0)

DEC R18 ; Decrement counter

BRNE LOOP ; Repeat 10 times

; 5. Store final result in R31:R30 as requested

MOV R30, R16

MOV R31, R17

HERE: RJMP HERE

ACTIVITY 6

Write a program to add two BCD numbers and store the result in **RAM** location $100 - $104. The two multibyte items are stored in the program memory starting at $120 as following data.

```
          .ORG  $120
DATA_1:   .DB   $54,$76,$65,$98    ;number 0x98657654
DATA_2:   .DB   $93,$56,$77,$38    ;number 0x38775693
```

INCLUDE "m328pdef.inc"

.CSEG

.ORG 0x120

DATA_1: .DB 0x54, 0x76, 0x65, 0x98 ; Little Endian: 0x98657654

DATA_2: .DB 0x93, 0x56, 0x77, 0x38 ; Little Endian: 0x38775693

.DSEG

.ORG 0x0100

RESULT_RAM: .BYTE 5 ; Reserve 5 bytes in SRAM

LAB 3-D

ARITHMETIC INSTRUCTIONS

```
.CSEG
.ORG 0x0000
 ; 1. Load Data 1 from Flash into Registers R16-R19
 LDI ZL, LOW(2 * DATA_1)
 LDI ZH, HIGH(2 * DATA_1)
 LPM R16, Z+
 LPM R17, Z+
 LPM R18, Z+
 LPM R19, Z+
 ; 2. Load Data 2 from Flash into Registers R20-R23
 ; Z already points to DATA_2 because it auto-incremented 4 times
 LPM R20, Z+
 LPM R21, Z+
 LPM R22, Z+
 LPM R23, Z+
LAB 3-D
ARITHMETIC INSTRUCTIONS
 ; 3. Perform 32-bit Addition (Result in R16-R19)
 ADD R16, R20 ; Add LSB
 ADC R17, R21 ; Add with Carry
 ADC R18, R22
 ADC R19, R23

 ; 4. Handle Final Carry (5th Byte)
 LDI R24, 0
 ADC R24, R24 ; R24 = 0 + 0 + C
```

LAB 3-D

ARITHMETIC INSTRUCTIONS

```
; 5. Store Result to SRAM starting at 0x100
 LDI YL, LOW(RESULT_RAM)
 LDI YH, HIGH(RESULT_RAM)


 ST Y+, R16 ; Store Byte 0
 ST Y+, R17 ; Store Byte 1
 ST Y+, R18 ; Store Byte 2
 ST Y+, R19 ; Store Byte 3
 ST Y+, R24 ; Store Byte 4 (Carry)
HALT: RJMP HALT
```