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
  ; Configure Ports
  LDI R16, 0x00
  OUT DDRC, R16      ; Port C as Input
  OUT DDRD, R16      ; Port D as Input
  LDI R16, 0xFF
  OUT DDRB, R16      ; Port B as Output

MAIN:
  ; 1. Read PORTC and Add 4
  IN R16, PINC      ; Read from PIN register for Input
  LDI R17, 4
  ADD R16, R17       ; R16 = PORTC + 4

  ; 2. Read PORTD
  IN R17, PIND      ; Read PORTD into R17

  ; 3. Multiply
  MUL R16, R17       ; Result in R1:R0 (R1=High, R0=Low)

  ; 4. Output Result
  OUT PORTB, R0      ; Send Lower Byte to PORTB

  RJMP MAIN
```

LAB 3-D

ARITHMETIC INSTRUCTIONS

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
    ; Configure Ports
    ; Assuming we read from PINB/PIND and write result to PORTB
    LDI R16, 0x00
    OUT DDRD, R16       ; Port D as Input

    ; Note: PORTB is used as both Input source and Output destination in the prompt.
    ; Usually, we configure it as Output to display, but reading PINB reads the
    ; actual state of the pins (which might be driven by external switches).
    ; For this code, we set PORTB as Output for the result.
    LDI R16, 0xFF
    OUT DDRB, R16

MAIN:
    IN R16, PINB        ; Read PORTB value
    IN R17, PIND        ; Read PORTD value

    ADD R16, R17        ; R16 = PORTB + PORTD

    LSR R16             ; Logical Shift Right (Divide by 2)
                        ; 0 -> b7 ... b0 -> C

    OUT PORTB, R16      ; Output result
    RJMP MAIN
```

ACTIVITY 3

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

```
         LDI   R16, 10
         LDI   R17, 20
         LDI   R18, 30
         MUL   R16, R17
         ADD   R0, R18
```

LAB 3-D

ARITHMETIC INSTRUCTIONS

3) 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
         LDI   R30, 30
         MUL   R30, R19
```

Code Snippet 1:

```
      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)
```

Therefore, R0: 230 (Decimal) or 0xE6 (Hex) and R1: 0 (Decimal) or 0x00 (Hex)

Code Snippet 2:

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

Thererfore, R0: 14 (Decimal) or 0x0E (Hex) (Lower byte of 270) and 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.
```

<u>Note</u> 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"

.CSEG
.ORG 0x200              ; Data located at word address $200
MYDATA: .DB 92, 34, 84, 129, 10, 20, 30, 40, 50, 60

.ORG 0x0000
    ; Initialize Z Pointer to byte address of MYDATA
    ; Flash is word-addressed, LPM uses byte address (Word * 2)
    LDI ZL, LOW(2 * MYDATA)
    LDI ZH, HIGH(2 * MYDATA)

    ; Initialize Sum Registers (R31:R30) to 0
    LDI R30, 0
    LDI R31, 0

    ; Initialize Loop Counter
    LDI R20, 10        ; 10 bytes to add
    LDI R21, 0         ; Register containing 0 for ADC

SUM_LOOP:
    LPM R16, Z+        ; Load byte from Flash into R16, Increment Z
    ADD R30, R16       ; Add to Lower Byte of Sum
    ADC R31, R21       ; Add Carry to Upper Byte of Sum

    DEC R20            ; Decrement Counter
    BRNE SUM_LOOP      ; Loop if not zero

HERE: RJMP HERE
```

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"

.CSEG
.ORG 0x300
MYDATA_BCD: .DB 0x92, 0x34, 0x84, 0x29, 0x10, 0x05, 0x01, 0x02, 0x03,
0x04

.ORG 0x0000
    ; Set Z Pointer to 0x300 * 2 = 0x600 (Byte Address)
    LDI ZL, LOW(0x600)
    LDI ZH, HIGH(0x600)

    ; Clear Sum (R31:R30) and Zero Reg (R21)
    LDI R30, 0
    LDI R31, 0
    LDI R21, 0
    LDI R20, 10         ; Counter

LOOP_BCD:
    LPM R16, Z+
    ADD R30, R16
    ADC R31, R21        ; Propagate carry
    DEC R20
    BRNE LOOP_BCD

DONE: RJMP DONE
```

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

.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

  ; 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
```