MICROPROCESSOR LAB EXPERIMENT 4

GROUP - 18
Deenabandhan N ee23b021
Sai Harshith Gajendra ee23b069
Krutarth Patel ee23b137

04 September 2024

Introduction :

- In this experiment, we are going to learn how to program the micro controller **ATmega8**.
- This experiment involves ,
  - Introduction to the assembly language.
  - Write a program in assembly language to display the maximum and minimum of 10 numbers stored in **FLASH** memory.
  - Write a program in assembly language to add 10 numbers stored in flash memory and store it in the register.
  - Sort 5 numbers stored in flash memory in arbitrary order and write the final results to data memory
- In this report , we have included the code of the tasks and our experience with the assembly language.

ATmega-8 and Microchip studio :

- Atmega-8 is an 8-bit RISC single-chip microcontroller developed by Atmel.
- The number 8 in its name represents that it can operate 8 bits at a time while processing the information i.e in a way it represents the capacity of the microcontroller.
- Some features of AVR microcontroller are
  - I/O ports.
  - Internal instructions flash memory
  - SRAM upto 16KB
  - Timers
- Flash memory is used to store the programs whatever we have written in the microchip studio.
- Each instruction will occupy the size of 2 bytes/16 bits in flash memory except for the instructions like **STS** , **JMP** which will occupy 4 bytes in the memory.
- For example the following code ,

```
LDI R16,0x01
```

  will occupy 2 bytes in the memory.
- Flash memory also has 32 registers (from R0 to R31) with three pointers ,
  - Z pointer : R30 and R31
  - Y pointer : R28 and R27
  - X pointer : R25 and R26
- These registers are used to hold memory in addition to having SRAM whose address starts from 0x60.
- We will see the instructions to implement the logic in the following sections.

Instructions used :

Instruction	Usage
LDI Rx,Rd	Load the value **Rd** in the register **Rx**
LD Rx,Rm	Load the value stored in the memory address **Rm** in the register **Rx**
LPM Rx,Rm	Load the value stored in the program memory address **Rm** in the register **Rx**
ST Rm,Rx	Store the address of the register **Rx** in the pointer **Rm**
MOV Rm1,Rm2	Copy the value stored in the register **Rm1** to the register **Rm2**
SUB Rm1,Rm2	Subtract the value stored in **Rm1** from the value stored in **Rm2** register and store the result in **Rm1**
CP Rm1,Rm2	Compare the values of the registers **Rm1** and **Rm2** and raise the following flags in the status registers - Sign flag - Negative flag - Carry flag if the first value is smaller than the second value.
DEC Rm	Decrease the value in the register **Rm** by an unit.

Finding the sum of numbers

Introduction :

- This task involves iterating through the registers and finding the sum
- It will always take n computations where n is the total number of values.

Code for Sum

```
.CSEG


LDI ZL, LOW(NUM<<1) ; Load the Z pointer with the array NUM
LDI ZH, HIGH(NUM<<1) ; Load the Z pointer with the array NUM

; R16 will always store the sum of the array
LPM R16, Z+ ; Load the first value of array

LDI R18, 0x09 ; Counter

LOOP :
        LPM R17,Z+ ; Load the next value to R17
        ADD R16,R17 ; Adding
        DEC R18 ; dereasing the counter
BRNE LOOP

NOP

NUM: .db 0x01,0x09,0x08,0x00,0x16,0x12,0x13,0x14,0x15,0x19
```

Usage of registers in flash memory in this code :

Registers	Usage
R18	Counter variable
R16 and **R17**	To store sum and temporary variable to store the loaded value
Z pointer	Store the array address to flash memory where the values are given.

Process

- Getting the values from the array and storing in a temporary variable.
- Adding the loaded value to the current sum
- Updating the sum

Finding the maximum and minimum number

Introduction :

- This task involves iterating through the registers and finding the minimum and maximum number
- It will always take n computations where n is the total number of values.

Code for minimum:

```
.CSEG

LDI ZL, LOW(NUM<<1) ; Load the Z pointer with the array NUM
LDI ZH, HIGH(NUM<<1) ; Load the Z pointer with the array NUM

; R16 will always store the minimum value of the array
LPM R16, Z+ ; Load the first value of array to min

LDI R18, 0x09 ; Counter

LOOP :
        LPM R17,Z+ ; Load the next value to R17
        CP R16,R17 ; Compare
        BRLO CDT ; This will go to CDT by skipping the next line if R16 < R17
        MOV R16,R17
        CDT:
                DEC R18 ; dereasing the counter
BRNE LOOP

NOP

NUM: .db 0x01,0x09,0x08,0x00,0x16,0x12,0x13,0x14,0x15,0x19
```

Code for maximum:

```
.CSEG

LDI ZL, LOW(NUM<<1) ; Load the Z pointer with the array NUM
LDI ZH, HIGH(NUM<<1) ; Load the Z pointer with the array NUM

; R16 will always store the maximum value of the array
LPM R16, Z+ ; Load the first value of array to max

LDI R18, 0x09 ; Counter

LOOP :
        LPM R17,Z+ ; Load the next value to R17
        CP R16,R17 ; Compare
        BRSH CDT ; This will go to CDT by skipping the next line if R16 > R17
        MOV R16,R17
        CDT:
                DEC R18 ; dereasing the counter
BRNE LOOP

NOP

NUM: .db 0x01,0x09,0x08,0x00,0x16,0x12,0x13,0x14,0x15,0x19
```

Usage of registers in flash memory in this code :

Registers	Usage
R18	Counter variable
R16 and **R17**	To store min or max and temporary variable to store the loaded value
Z pointer	Store the array address where the values are given.

Process

- Getting the values from the array and storing in a temporary variable.
- Comparing with current min or max with the new value from the array
- Changing min or max if required or restarting the loop with a decrease in the counter variable

Sorting the stored numbers in the Flash memory

Introduction :

- This task involves the knowledge of iterating through the registers multiple times and comparing the values in the given memory addresses.
- We have implemented the bubble sort algorithm to sort the array.
- At the worst case (numbers are in descending order), it will take $n*(n-1)/2$ computations to implement sorting.

Code :

```
.CSEG

LDI ZL,LOW(NUM<<1) ; Load the Z pointer with the array NUM
LDI ZH,HIGH(NUM<<1) ; Load the Z pointer with the array NUM

LDI YL,LOW(0x60) ; Load the Y pointer with the register stored in SRAM
LDI YH,HIGH(0x60) ; Load the Y pointer with the register stored in SRAM

LPM R25,Z+ ; Storing the first value of the array in R25
ST Y+,R25 ; Storing R25 into the register in SRAM

LDI R22,0x04 ; R22 will store the number of elements to intake from NUM (5-1)

LOOP :
        LDI R23,0x05 ; R23 will store the number of sortings that it has to do
        SUB R23,R22 ; R23 will be 5 - Current iteration i.e R22

        LPM R25,Z+ ; Storing the first value of the array in R25
        ST Y+,R25 ; Storing R25 into the register in SRAM

        MOV XL,YL ; Creating a X pointer to iterate through the sorting process
        MOV XH,YH ; Copying the current address stored in Y pointer in the X pointer

        LD R25,-X ; To shift to the last stored elements
        LOOP1 :
                LD R25,X ; Loading the current value stored at X to R25
                LD R24,-X ; Loading the current value stored at -X to R25

                CP R24,R25; Comparing the values
                BRLO CDT ; This will skip the next lines and jump to CDT if R24 < R25 (No swapping require

                ST X+,R25 ; Swapping current position with R25
                ST X,R24 ; Swapping right adjacent position with R24
                LD R24,-X ; Moving back to the left adjacent position inorder to continue the loop

                CDT:
                DEC R23 ; Decreasing thr counter
        BRNE LOOP1 ; Running through the loop
        DEC R22 ; Decreasing thr counter
BRNE LOOP ; Running through the loop

NOP
NUM: .db 0x01,0x11,0x08,0x05,0x02
```

Usage of registers in flash memory in this code :

Registers	Usage
R22	Counter variable to iterate through the list.
R23	Counter variable to reiterate through the stored elements to check for swapping.
R24 and **R25**	Temporary registers to store the values at pointers and used to swap if necessary.
Z pointer	Store the array address where the values are given.
Y pointer	Store the SRAM register address so that we can iterate through the contiguous registers to store the values.
X pointer	Store the current Y pointer so that we can iterate backwards to check for swapping.

Processes :

- Let us analyse the above code in terms of three processes.
  - Getting the values from the array.
  - Iterating backward to check if we have to swap.
  - Swapping condition.

Getting the values from the array :

- The register **R25** is used as a temporary register to assign the value at **Z** to the register which is at the memory **Y**.
- The code which do this process is

```
LDI ZL,LOW(NUM<<1)
LDI ZH,HIGH(NUM<<1)

LDI YL,LOW(0x60)
LDI YH,HIGH(0x60)

LPM R25,Z+
ST Y+,R25
```

Iterating backward to check for swapping :

- We would need an another pointer to store the current pointer values so that we can back propagate which is done by,

```
MOV XL,YL
MOV XH,YH
```

- But Y pointer will store the address of the next register to be used. So we should iterate back once before entering the loop which is done by

```
LD R25,-X
```

- The number of times that the loop should back propagate is determined by the number of elements stored which is $\boxed{5 - \mathbf{R22}}$ which is done by,

```
LDI R23,0x05
SUB R23,R22
```

Swapping condition :

- We have to check the values stored in the registers at memory locations of the current position and the previous position which is done by ,

```
LD R25,X
LD R24,-X
CP R24,R25
```

- If the first value is greater than the second one , we have to swap the values else nothing should be done which is represented as ,

```
BRLO CDT

ST X+,R25
ST X,R24
LD R24,-X

CDT:
   DEC R23
```

Result :