Experiment 2: Computations using Atmel Atmega8 AVR through Assembly Program Emulation

Santhosh S P ee21b119

1 Addition of two 8-bit numbers:

1.1 Flowchart for the problem

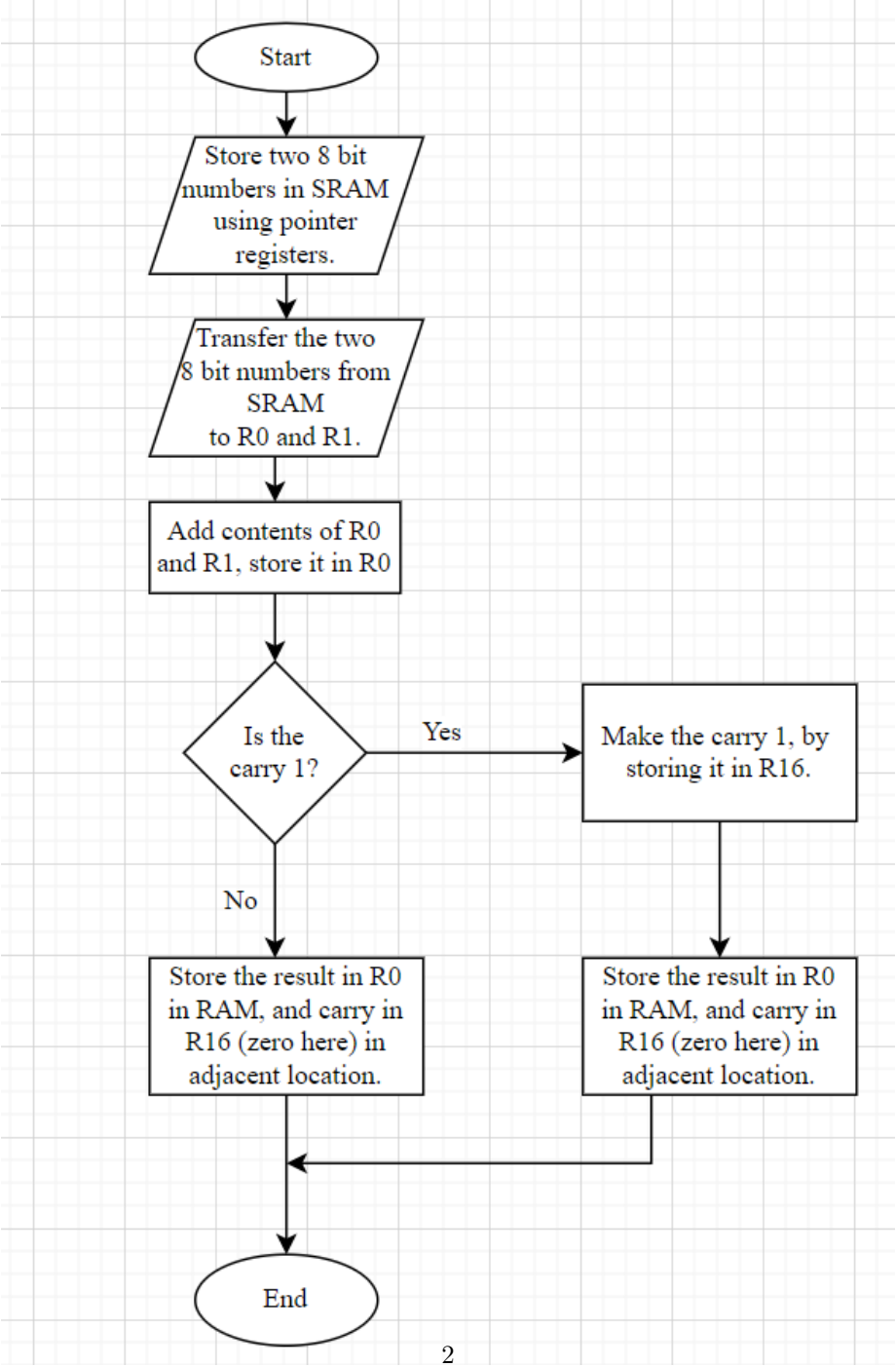

Figure 1: Flowchart for the problem

1.2 Code

```
 .CSEG ; define memory space to hold program - code segment
LDI ZL,LOW (NUM << 1);load byte address (not word address)of
LDI ZH,HIGH (NUM << 1); first data byte (first number)
LDI XL,0x60;load SRAM address in X-register
LDI XH,0x00; MSB byte
LDI R16,00; clear R16, used to hold carry
LPM R0,Z+; Z now points to a single byte - first number
LPM R1,Z; Get second number into R1
ADD R0,R1; Add R0 and R1,result in R0,carry flag affected
BRCC abc; jump if no carry,
LDI R16,0x01 ; else make carry 1
abc: ST X+,R0 ; store result in RAM
ST X,R16 ; store carry in next location
NOP ; End of program, No operation
NUM: .db 0xDD,0xFF;
```

1.3 Output

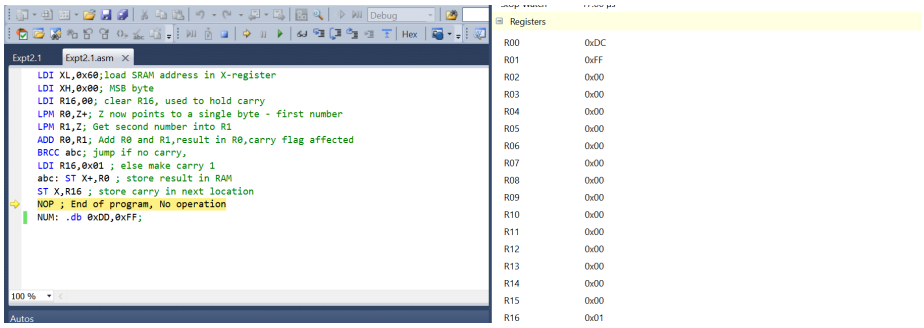

Figure 2: 0xDD+0xFF = 0x1DC. The carry bit is stored in R16, and the remaining 8 bits are stored in R0.

2 16-bit addition using 8-bit processor:

2.1 Flowchart for the problem

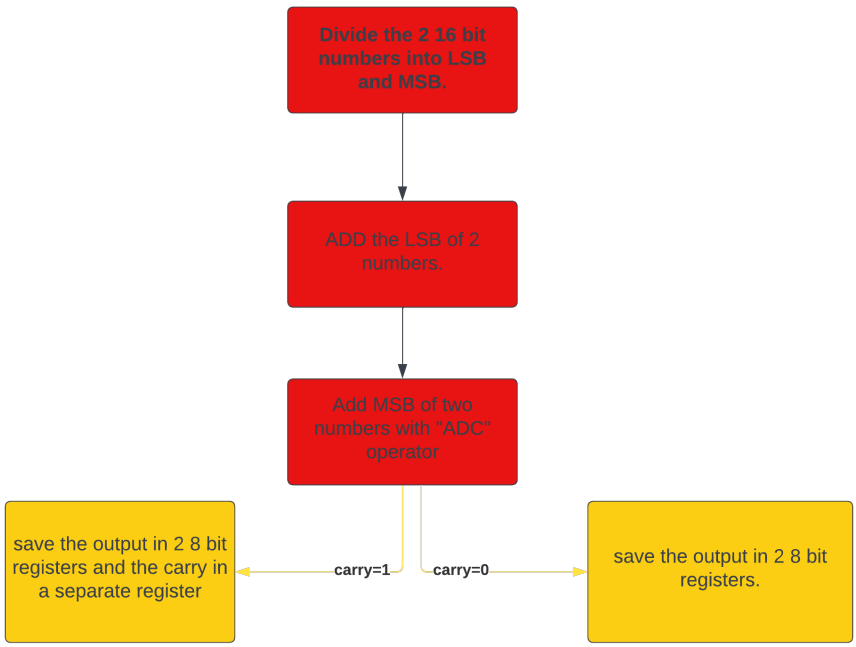

Figure 3: Flowchart for the problem

2.2 Code

```
 .CSEG; define memory space to hold program
LDI ZL, LOW  ( NUM1 << 1);load byte address (not word address)of
LDI ZH, HIGH ( NUM1 << 1); first data byte (first number)
LDI R17,0x00;clear R17, used to hold the carry of the sum of LSB 8 bits
LDI R16,0x00;clear R16, used to hold overflow
LPM R0,Z+; Z now points to MSB of first number
LPM R1,Z+; Z now points to LSB of second number
LPM R2,Z+; Z now points to MSB of second number
LPM R3,Z
ADD R1,R3
BRCC abc; jump if no carry,
LDI R17,0x01 ; else make carry 1
ADC R0,R2
```

```
BRCC pqr
LDI R16,0x01
abc:ADC R0,R2;
pqr:NOP ; end

NUM1: .db 0xDE,0xFF,0xC5,0xE2 ;numbers to be added 0xDEFF and 0xC5E2
```

2.3 Output

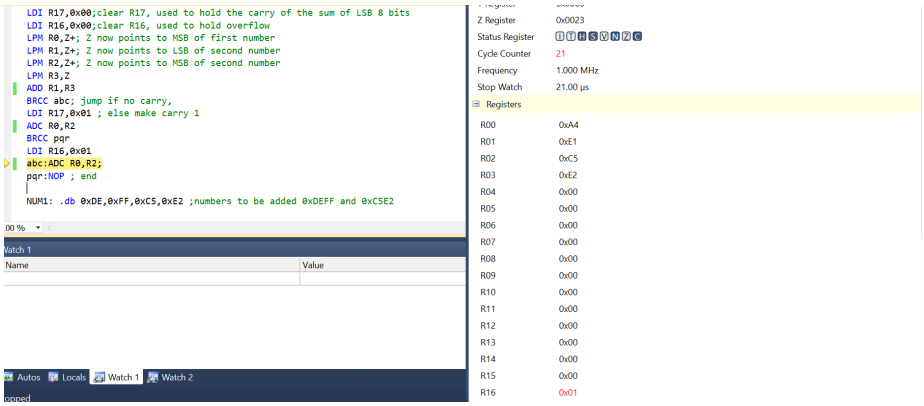

Figure 4: 0xDEFF + 0xC5E2 = 0x1A4E1. The overflow bit (MSB) is stored in R16. The next 8 bits are stored in R1, and the remaining 8 bits are stored in R0.

3 Multiplication of two 8-bit numbers:

3.1 Flowchart for the problem

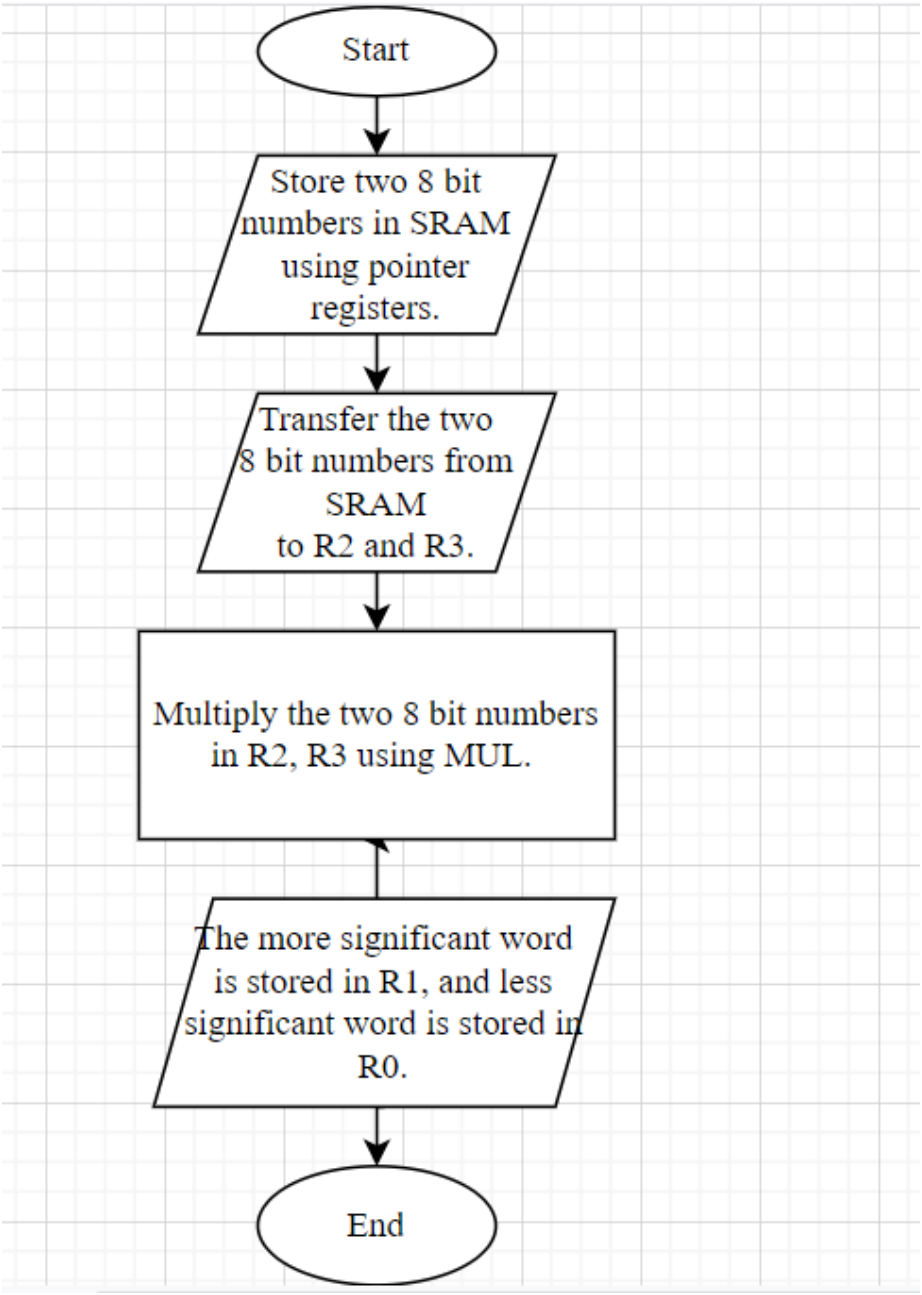

Figure 5: Flowchart for the problem

3.2 Code

```
.CSEG; define memory space to hold program
LDI ZL, LOW  ( NUM1 << 1);load byte address
LDI ZH, HIGH ( NUM1 << 1); first data byte (first number)
LPM R2,Z+; Z now points to second number
LPM R3,Z
MUL R2,R3
NOP ; End
NUM1: .db 0x27,0x12;
```

3.3 Output

Figure 6: 0x27*0x12 = 0x2BE. The first 8 bits (most significant ones) are stored in R1 while the remaining 8 bits are stored in R0.

4 Largest number in a given set:

4.1 Flowchart for the problem

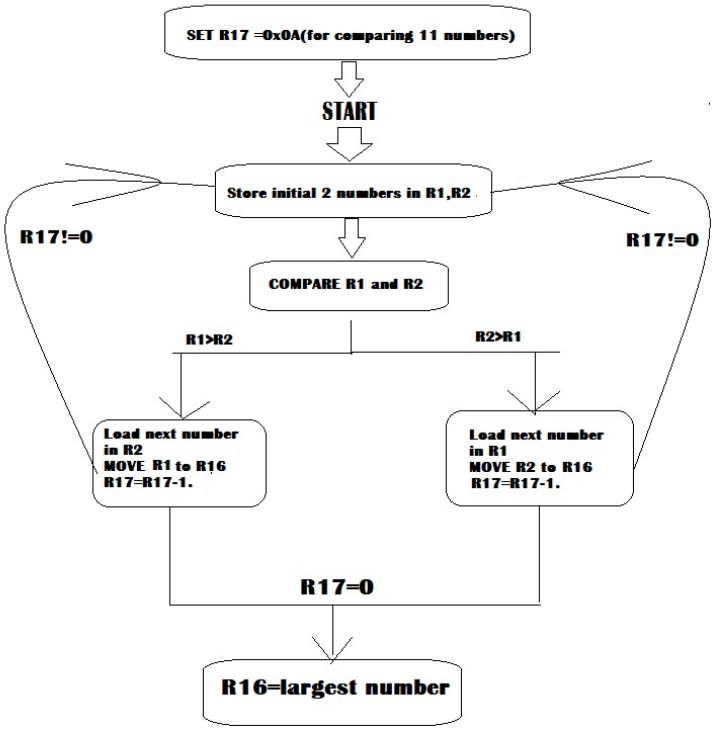

Figure 7: Flowchart for the problem

4.2 Code

```
.CSEG; define memory space to hold program - code segment
LDI ZL, LOW  ( NUM1 << 1) ;load byte address (not word address)of
LDI ZH, HIGH ( NUM1 << 1) ; first data byte (first number)
LDI R17,0x0A
DEC R17
LDI R16,0x00
LPM R1,Z+ ; Z now points to LSB of first number
LPM R2,Z+ ; Z now points to MSB of second number

do: CP R1,R2
   BRSH r1r2
```

```
   BRLO r2r1
   r2r1: LPM R1,Z+
MOV R16,R2
DEC R17
        BRNE do
NOP
   r1r2: LPM R2,Z+
MOV R16,R1
DEC R17
        BRNE do
NOP

NUM1: .db 0x00,0x07,0x09,0x018,0x11,0x29,0x00,0x22,0x33,0x34
```

4.3 Output

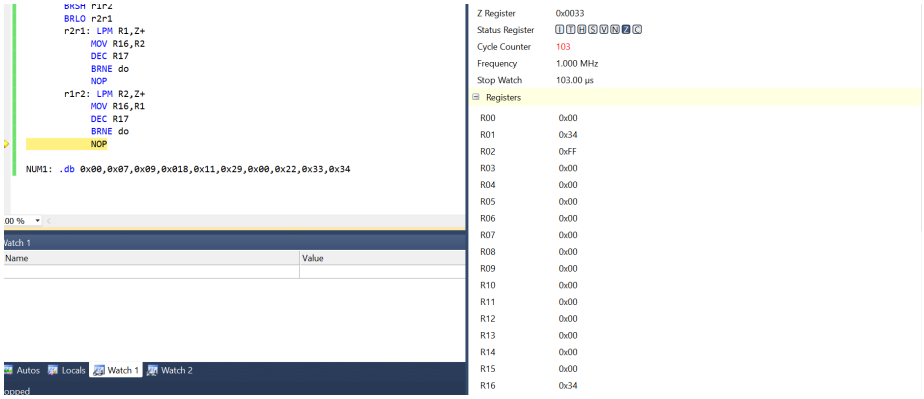

Figure 8: The largest number in the given group is stored in R16.