Elizabeth Heider

CPE301 – SPRING 2018

Design Assignment 1

DO NOT REMOVE THIS PAGE DURING SUBMISSION:

The student understands that all required components should be submitted in complete for grading of this assignment.

NO	SUBMISSION ITEM	COMPLETED (Y/N)	MARKS (/MAX)
1	COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS		
2.	INITIAL CODE OF TASK 1/A		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 3/C		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 4/D		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 5/E		
4.	SCHEMATICS		
5.	SCREENSHOTS OF EACH TASK OUTPUT		
5.	SCREENSHOT OF EACH DEMO		
6.	VIDEO LINKS OF EACH DEMO		
7.	GOOGLECODE LINK OF THE DA		

1) **INITIAL CODE OF TASK 1:** Store 300 numbers starting from the STARTADDS=0x0222 location. Populate the value of the memory location by adding high(STARTADDS) and low(STARTADDS). Use X/Y/Z Registers as pointers to fill up 300 numbers.

```
LDI R16, HIGH(RAMEND) // Intializing Stack Pointer
OUT SPH, R16
LDI R16, LOW(RAMEND)
OUT SPL, R16

LDI R26, 0x22; store 0x22 into X-Low
LDI R27, 0x02; store 0x02 into X-High

LDI R28, 0x00 ; store 0x0400 into Y Low
LDI R29, 0x04 ; store 0x0400 into Y High

LDI R30, 0x00 ; store 0x0600 into Z Low
LDI R31, 0x06 ; store 0x0600 into Z High

LDI R20, 0x30; // Counter1 Value 50 in Dec (Branch options only work up to 60)
LDI R22, 0x5; // Counter to RELOAD Counter1 (50*6=300)

MOV R23, R26  ; copy R26 into R23 to not change value of R26

START:
ADD R23, R27 ; Add High and Low of Address
ST X+, R23; Store Sum into Address, Increment X as the pointer
MOV R21, R23; Copy R23 into R21 to use R21 for subtraction/loop
```

2) **INITIAL CODE OF TASK 2:** Use X/Y/Z Register addressing to parse through 300 numbers, if the number is divisible by 5 store the number from memory location 0x0400, else store at location starting 0x0600.

```
CHECK:
SUBI R21, 0x5; Subtract 5 from R21

CPI R21, 0x00; Compare R21 to Zero
BRLT NOTDIV; If R21 < 0 number NOT divisible by 5 / Go to function

CPI R21, 0X00; Compare R21 to zero
BREQ DIV; if R21 = 0 Number is divisible by 5 / Go to function

RJMP CHECK; No conditions met, jump to CHECK and restart loop

DIV: // Function to store Numbers Divisible by 5
ST Y+, R23; Store value of R23 into addr, inc pointer value
ADD R16, R23 ; add low bit to R16
ADC R17, R0 ; add carry to R17/High
RJMP END ; Jump to end

NOTDIV: // Function to store numbers not divisible by 5
ST Z+, R23; store value into addr, inc pointer value
ADD R18, R23 ; add low bit to R16
ADC R19, R0 ; add carry
RJMP END ; jump to end
```

3) **INITIAL CODE OF TASK 2:** Use X/Y/Z Register addressing to simultaneously add numbers from memory location 0x0400 and 0x0600 and store the sums at R16:R17 and R18:R19 respectively.

```
DIV: // Function to store Numbers Divisible by 5
ST Y+, R23; Store value of R23 into addr, inc pointer value
ADD R16, R23 ; add low bit to R16
ADC R17, R0 ; add carry to R17/High
RJMP END ; Jump to end

NOTDIV: // Function to store numbers not divisible by 5
ST Z+, R23; store value into addr, inc pointer value
ADD R18, R23 ; add low bit to R16
ADC R19, R0 ; add carry
RJMP END ; jump to end
```

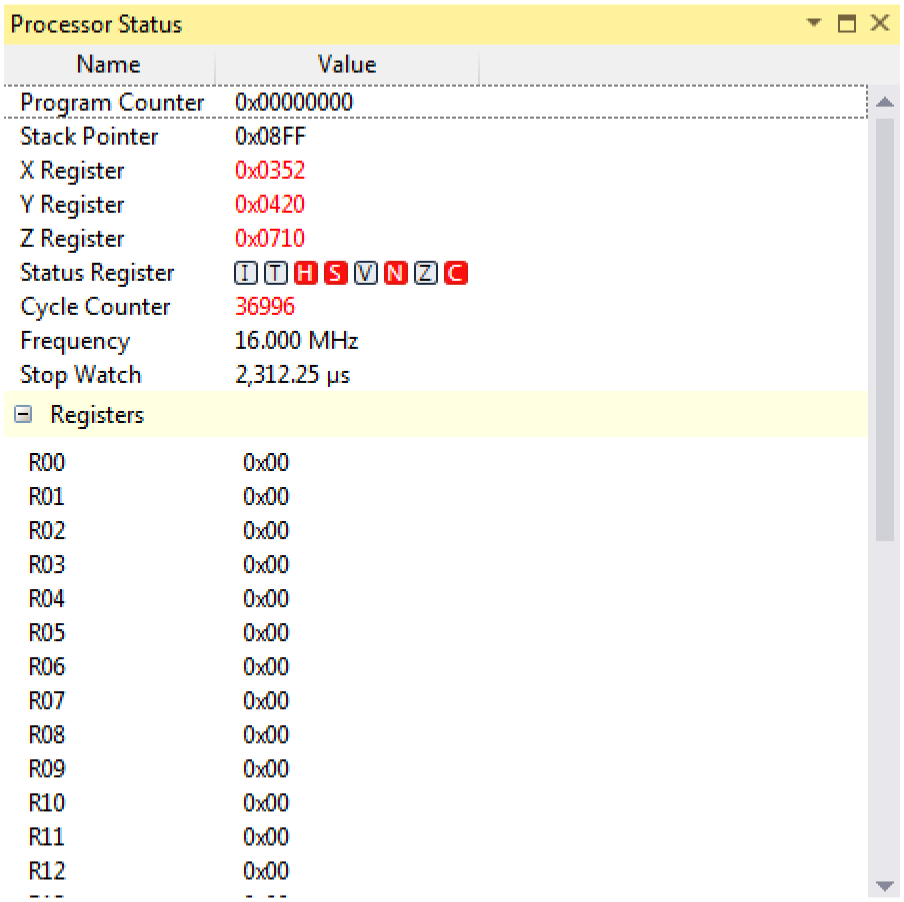
4) Complete Code

```
LDI R16, HIGH(RAMEND) // Intializing Stack Pointer
OUT SPH, R16
LDI R16, LOW(RAMEND)
OUT SPL, R16

LDI R26, 0x22; store 0x22 into X-Low
LDI R27, 0x02; store 0x02 into X-High

LDI R28, 0x00 ; store 0x0400 into Y Low
LDI R29, 0x04 ; store 0x0400 into Y High

LDI R30, 0x00 ; store 0x0600 into Z Low
LDI R31, 0x06 ; store 0x0600 into Z High

LDI R20, 0x30; // Counter1 Value 50 in Dec (Branch options only work up to 60)
LDI R22, 0x5; // Counter to RELOAD Counter1 (50*6=300)

MOV R23, R26  ; copy R26 into R23 to not change value of R26

START:
ADD R23, R27 ; Add High and Low of Address
ST X+, R23; Store Sum into Address, Increment X as the pointer
MOV R21, R23; Copy R23 into R21 to use R21 for subtraction/loop

CHECK:
SUBI R21, 0x5; Subtract 5 from R21
CPI R21, 0x00; Compare R21 to Zero
BRLT NOTDIV; If R21 < 0 number NOT divisible by 5 / Go to function
CPI R21, 0X00; Compare R21 to zero
BREQ DIV; if R21 = 0 Number is divisible by 5 / Go to function

RJMP CHECK; No conditions met, jump to CHECK and restart loop

DIV: // Function to store Numbers Divisible by 5
ST Y+, R23; Store value of R23 into addr, inc pointer value
ADD R16, R23 ; add low bit to R16
ADC R17, R0 ; add carry to R17/High
RJMP END ; Jump to end

NOTDIV: // Function to store numbers not divisible by 5
ST Z+, R23; store value into addr, inc pointer value
ADD R18, R23 ; add low bit to R16
ADC R19, R0 ; add carry
RJMP END ; jump to end

END:
SUBI R20, 0x01; /subtracting from counter
BRPL START ; if counter > 0 branch to start

LDI R20, 0x32;  "load" counter with value 50
SUBI R22, 0x01; counter for reload amount
BRPL START ; loop to start

DONE:
```

6) C-Program for Verification of Values

```
int main()
{

	int *x = 0x0222; // * indication of pointers
	int *y = 0x0400;
	int *z = 0x0600;


	int i = 0;
	int addr = 0;

	char sum1 = 0;
	char sum2 = 0;

	char R16;
	char R17;
	char R18;
	char R19;


}
	for (i=0; i < 300; i++)
	{


		sum1 = 0;
		sum2 = 0;

		*x = addr;
		if (addr/5 == 0 )
		{
			*Y = sum;
			Y++;
			R16 = sum1;
			R17 - sum2;

		}

		else
		{
			*z = sum;
			R18 = sum1;
			R19 = sum2;

		}
		x = x + 0x02;
	}
```

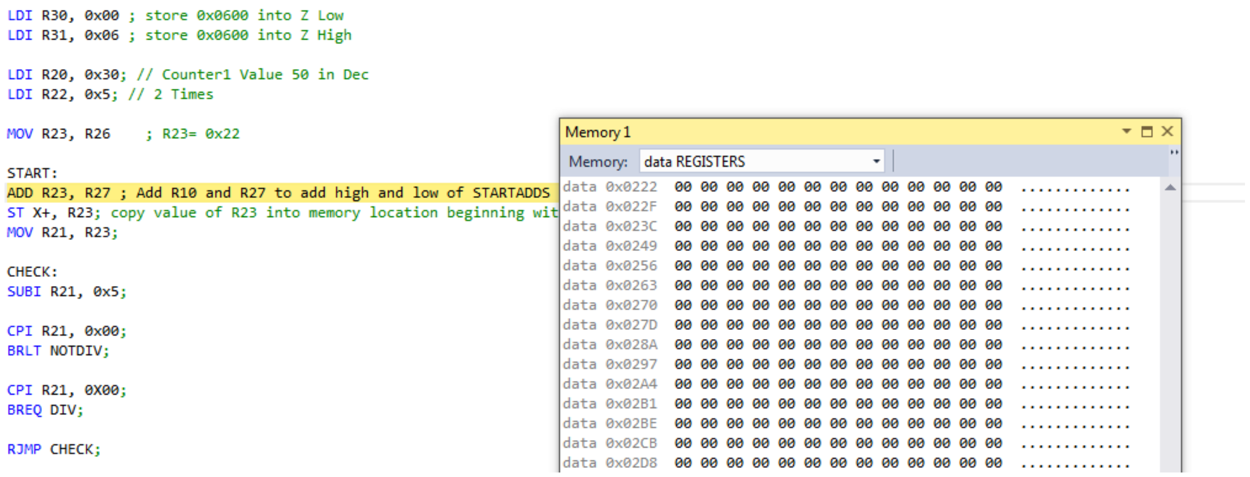
5) **Determine the Execution Time @ 16 MHz/Cycles**



Processor Status

Name	Value
Program Counter	0x00000000
Stack Pointer	0x08FF
X Register	0x0352
Y Register	0x0420
Z Register	0x0710
Status Register	I T H S V N Z C
Cycle Counter	36996
Frequency	16.000 MHz
Stop Watch	2,312.25 µs
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00
R08	0x00
R09	0x00
R10	0x00
R11	0x00
R12	0x00

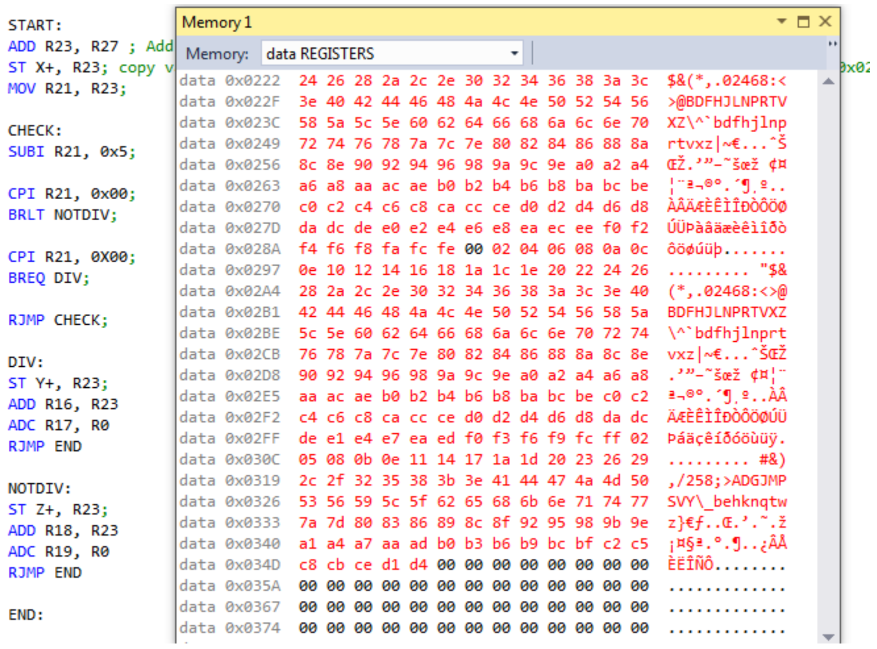
Clock at 16MHz, execution done in 2,312 microseconds.



```
LDI R30, 0x00 ; store 0x0600 into Z Low
LDI R31, 0x06 ; store 0x0600 into Z High

LDI R20, 0x30; // Counter1 Value 50 in Dec
LDI R22, 0x5; // 2 Times

MOV R23, R26    ; R23= 0x22

START:
ADD R23, R27 ; Add R10 and R27 to add high and low of STARTADDS
ST X+, R23; copy value of R23 into memory location beginning wit
MOV R21, R23;

CHECK:
SUBI R21, 0x5;

CPI R21, 0x00;
BRLT NOTDIV;

CPI R21, 0X00;
BREQ DIV;

RJMP CHECK;
```

```
Memory 1
Memory:  data REGISTERS
data 0x0222  00 00 00 00 00 00 00 00 00 00 00 00 00  .............
data 0x022F  00 00 00 00 00 00 00 00 00 00 00 00 00  .............
data 0x023C  00 00 00 00 00 00 00 00 00 00 00 00 00  .............
data 0x0249  00 00 00 00 00 00 00 00 00 00 00 00 00  .............
data 0x0256  00 00 00 00 00 00 00 00 00 00 00 00 00  .............
data 0x0263  00 00 00 00 00 00 00 00 00 00 00 00 00  .............
data 0x0270  00 00 00 00 00 00 00 00 00 00 00 00 00  .............
data 0x027D  00 00 00 00 00 00 00 00 00 00 00 00 00  .............
data 0x028A  00 00 00 00 00 00 00 00 00 00 00 00 00  .............
data 0x0297  00 00 00 00 00 00 00 00 00 00 00 00 00  .............
data 0x02A4  00 00 00 00 00 00 00 00 00 00 00 00 00  .............
data 0x02B1  00 00 00 00 00 00 00 00 00 00 00 00 00  .............
data 0x02BE  00 00 00 00 00 00 00 00 00 00 00 00 00  .............
data 0x02CB  00 00 00 00 00 00 00 00 00 00 00 00 00  .............
data 0x02D8  00 00 00 00 00 00 00 00 00 00 00 00 00  .............
```

Task 1 Before Execution

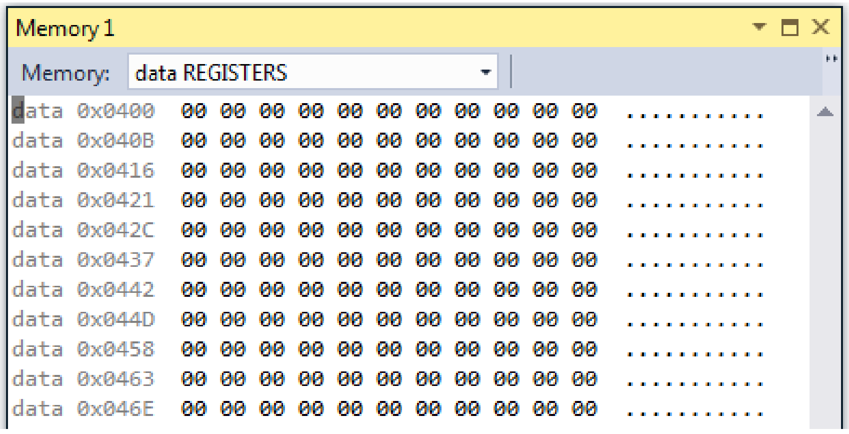


```
START:
ADD R23, R27 ; Add
ST X+, R23; copy v
MOV R21, R23;

CHECK:
SUBI R21, 0x5;

CPI R21, 0x00;
BRLT NOTDIV;

CPI R21, 0X00;
BREQ DIV;

RJMP CHECK;

DIV:
ST Y+, R23;
ADD R16, R23
ADC R17, R0
RJMP END

NOTDIV:
ST Z+, R23;
ADD R18, R23
ADC R19, R0
RJMP END

END:
```

```
Memory 1
Memory:  data REGISTERS
data 0x0222  24 26 28 2a 2c 2e 30 32 34 36 38 3a 3c  $&(*,.02468:<
data 0x022F  3e 40 42 44 46 48 4a 4c 4e 50 52 54 56  >@BDFHJLNPRTV
data 0x023C  58 5a 5c 5e 60 62 64 66 68 6a 6c 6e 70  XZ\^`bdfhjlnp
data 0x0249  72 74 76 78 7a 7c 7e 80 82 84 86 88 8a  rtvxz|~€...ˆŠ
data 0x0256  8c 8e 90 92 94 96 98 9a 9c 9e a0 a2 a4  ŒŽ.’”–˜šœž ¢¤
data 0x0263  a6 a8 aa ac ae b0 b2 b4 b6 b8 ba bc be  ¦¨ª¬®°.´¶¸º..
data 0x0270  c0 c2 c4 c6 c8 ca cc ce d0 d2 d4 d6 d8  ÀÂÄÆÈÊÌÎÐÒÔÖØ
data 0x027D  da dc de e0 e2 e4 e6 e8 ea ec ee f0 f2  ÚÜÞàâäæèêìîðò
data 0x028A  f4 f6 f8 fa fc fe 00 02 04 06 08 0a 0c  ôöøúüþ.......
data 0x0297  0e 10 12 14 16 18 1a 1c 1e 20 22 24 26  ......... "$&
data 0x02A4  28 2a 2c 2e 30 32 34 36 38 3a 3c 3e 40  (*,.02468:<>@
data 0x02B1  42 44 46 48 4a 4c 4e 50 52 54 56 58 5a  BDFHJLNPRTVXZ
data 0x02BE  5c 5e 60 62 64 66 68 6a 6c 6e 70 72 74  \^`bdfhjlnprt
data 0x02CB  76 78 7a 7c 7e 80 82 84 86 88 8a 8c 8e  vxz|~€...ˆŠŒŽ
data 0x02D8  90 92 94 96 98 9a 9c 9e a0 a2 a4 a6 a8  .’”–˜šœž ¢¤¦¨
data 0x02E5  aa ac ae b0 b2 b4 b6 b8 ba bc be c0 c2  ª¬®°.´¶¸º..ÀÂ
data 0x02F2  c4 c6 c8 ca cc ce d0 d2 d4 d6 d8 da dc  ÄÆÈÊÌÎÐÒÔÖØÚÜ
data 0x02FF  de e1 e4 e7 ea ed f0 f3 f6 f9 fc ff 02  Þáäçêíðóöùüÿ.
data 0x030C  05 08 0b 0e 11 14 17 1a 1d 20 23 26 29  ......... #&)
data 0x0319  2c 2f 32 35 38 3b 3e 41 44 47 4a 4d 50  ,/258;>ADGJMP
data 0x0326  53 56 59 5c 5f 62 65 68 6b 6e 71 74 77  SVY\_behknqtw
data 0x0333  7a 7d 80 83 86 89 8c 8f 92 95 98 9b 9e  z}€ƒ..Œ.’.˜.ž
data 0x0340  a1 a4 a7 aa ad b0 b3 b6 b9 bc bf c2 c5  ¡¤§ª.°.¶..¿ÂÅ
data 0x034D  c8 cb ce d1 d4 00 00 00 00 00 00 00 00  ÈËÎÑÔ........
data 0x035A  00 00 00 00 00 00 00 00 00 00 00 00 00  .............
data 0x0367  00 00 00 00 00 00 00 00 00 00 00 00 00  .............
data 0x0374  00 00 00 00 00 00 00 00 00 00 00 00 00  .............
```

Task 1 Post Execution

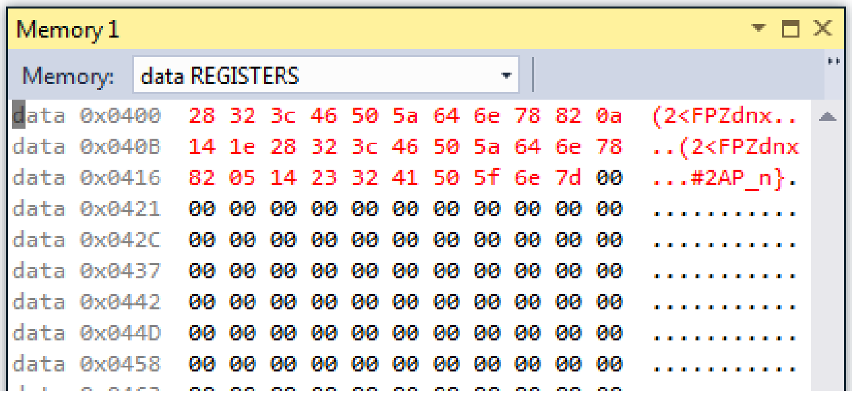
```
RJMP CHECK;

DIV:
ST Y+, R23;
ADD R16, R23
ADC R17, R0
RJMP END

NOTDIV:
ST Z+, R23;
ADD R18, R23
ADC R19, R0
RJMP END
```



```
Memory 1
Memory: data REGISTERS
data 0x0400  00 00 00 00 00 00 00 00 00 00 00  ...........
data 0x040B  00 00 00 00 00 00 00 00 00 00 00  ...........
data 0x0416  00 00 00 00 00 00 00 00 00 00 00  ...........
data 0x0421  00 00 00 00 00 00 00 00 00 00 00  ...........
data 0x042C  00 00 00 00 00 00 00 00 00 00 00  ...........
data 0x0437  00 00 00 00 00 00 00 00 00 00 00  ...........
data 0x0442  00 00 00 00 00 00 00 00 00 00 00  ...........
data 0x044D  00 00 00 00 00 00 00 00 00 00 00  ...........
data 0x0458  00 00 00 00 00 00 00 00 00 00 00  ...........
data 0x0463  00 00 00 00 00 00 00 00 00 00 00  ...........
data 0x046E  00 00 00 00 00 00 00 00 00 00 00  ...........
```

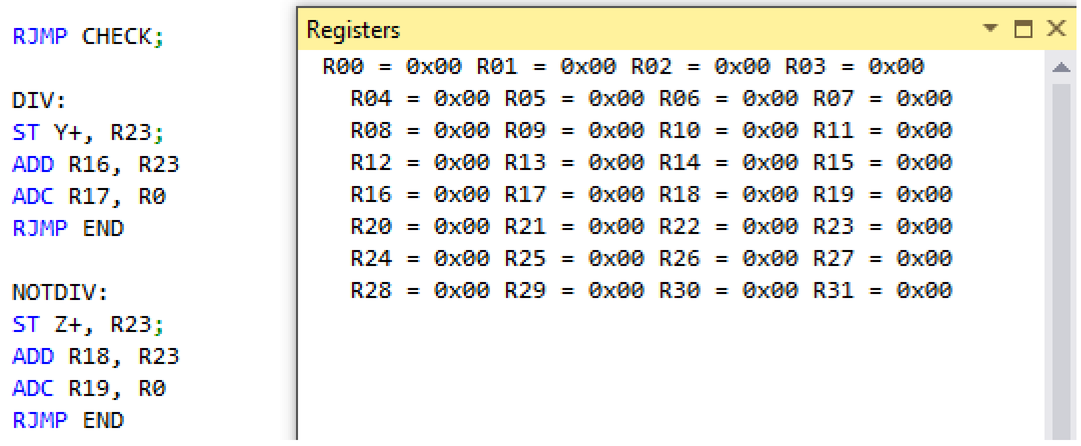
Task 2 Before Execution

```
CHECK:
SUBI R21, 0x5;

CPI R21, 0x00;
BRLT NOTDIV;

CPI R21, 0X00;
BREQ DIV;

RJMP CHECK;

DIV:
ST Y+, R23;
ADD R16, R23
ADC R17, R0
RJMP END

NOTDIV:
ST Z+, R23;
ADD R18, R23
ADC R19, R0
RJMP END
```



```
Memory 1
Memory: data REGISTERS
data 0x0400  28 32 3c 46 50 5a 64 6e 78 82 0a  (2<FPZdnx..
data 0x040B  14 1e 28 32 3c 46 50 5a 64 6e 78  ..(2<FPZdnx
data 0x0416  82 05 14 23 32 41 50 5f 6e 7d 00  ...#2AP_n}.
data 0x0421  00 00 00 00 00 00 00 00 00 00 00  ...........
data 0x042C  00 00 00 00 00 00 00 00 00 00 00  ...........
data 0x0437  00 00 00 00 00 00 00 00 00 00 00  ...........
data 0x0442  00 00 00 00 00 00 00 00 00 00 00  ...........
data 0x044D  00 00 00 00 00 00 00 00 00 00 00  ...........
data 0x0458  00 00 00 00 00 00 00 00 00 00 00  ...........
```

Task 2 Post Execution

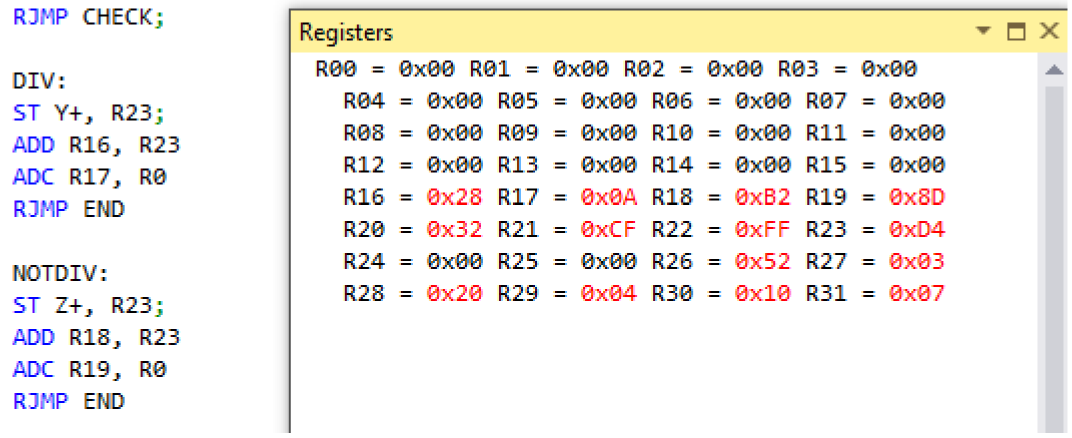


```
RJMP CHECK;

DIV:
ST Y+, R23;
ADD R16, R23
ADC R17, R0
RJMP END

NOTDIV:
ST Z+, R23;
ADD R18, R23
ADC R19, R0
RJMP END
```

Registers

```
R00 = 0x00 R01 = 0x00 R02 = 0x00 R03 = 0x00
  R04 = 0x00 R05 = 0x00 R06 = 0x00 R07 = 0x00
  R08 = 0x00 R09 = 0x00 R10 = 0x00 R11 = 0x00
  R12 = 0x00 R13 = 0x00 R14 = 0x00 R15 = 0x00
  R16 = 0x00 R17 = 0x00 R18 = 0x00 R19 = 0x00
  R20 = 0x00 R21 = 0x00 R22 = 0x00 R23 = 0x00
  R24 = 0x00 R25 = 0x00 R26 = 0x00 R27 = 0x00
  R28 = 0x00 R29 = 0x00 R30 = 0x00 R31 = 0x00
```

Task 3 Before Execution



```
RJMP CHECK;

DIV:
ST Y+, R23;
ADD R16, R23
ADC R17, R0
RJMP END

NOTDIV:
ST Z+, R23;
ADD R18, R23
ADC R19, R0
RJMP END
```

Registers

```
R00 = 0x00 R01 = 0x00 R02 = 0x00 R03 = 0x00
  R04 = 0x00 R05 = 0x00 R06 = 0x00 R07 = 0x00
  R08 = 0x00 R09 = 0x00 R10 = 0x00 R11 = 0x00
  R12 = 0x00 R13 = 0x00 R14 = 0x00 R15 = 0x00
  R16 = 0x28 R17 = 0x0A R18 = 0xB2 R19 = 0x8D
  R20 = 0x32 R21 = 0xCF R22 = 0xFF R23 = 0xD4
  R24 = 0x00 R25 = 0x00 R26 = 0x52 R27 = 0x03
  R28 = 0x20 R29 = 0x04 R30 = 0x10 R31 = 0x07
```

Task 3 Post Execution