Noor Blum

CPE301 – SPRING 2016

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)** |
| 0. | COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS | N/A | N/A |
| 1. | INITIAL CODE OF TASK 1/A | Y |  |
| 2. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B | Y |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 3/C | Y |  |
| 4. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 4/D | Y |  |
| 5. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 5/E | Y |  |
| 6. | FLOW DIAGRAM | Y |  |
| 7. | SCREENSHOTS OF EACH TASK OUTPUT | Y |  |
| 8. | SCREENSHOT OF EACH DEMO | Y |  |
| 9. | INSTRUCTION DOCUMENT | Y |  |
| 10. | GITHUB LINK THE DA1 | Y |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 0. | COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS |  |  |

* AVR Studio 6
* AVR Atmega 328 Development Board

|  |  |  |  |
| --- | --- | --- | --- |
| 1. | INITIAL CODE OF TASK 1/A |  |  |

/\*

\* CPE\_301\_Design\_Assignment\_a.asm

\* Created: 2/18/2016 4:12:28 PM

\* Author: blumn

\*/

.nolist

.include "m328pdef.inc"

.equ RAMSTART=0x0100

.list

.dseg

.cseg

.org $000

; init counter

ldi r16 , $19

; init pointer x to RAM middle

ldi r26 , low((RAMEND+RAMSTART)/2)

; init pointer y to RAM end

ldi r28 , low(RAMEND)

ldi r29 , high(RAMEND)

; loop

st Y, r26

memory\_copy\_loop:

dec r26

st -Y, r26

dec r16

brne memory\_copy\_loop

|  |  |  |  |
| --- | --- | --- | --- |
| 2. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B |  |  |

/\*

\* CPE\_301\_Design\_Assignment\_b.asm

\* Created: 2/18/2016 4:22:02 PM

\* Author: blumn

\*/

.nolist

.include "m328pdef.inc"

.equ stacktop = $08DF

.equ div\_no = $7

.list

.dseg

.cseg

.org $000

; init counter

ldi r16 , $19

; init stack pointer

ldi r22 , high(stacktop)

out SPH , r22

ldi r22 , low(stacktop)

out SPL , r22

; init pointer y to RAM end

ldi r28 , low(RAMEND)

ldi r29 , high(RAMEND)

; loop

memory\_parse\_loop:

ld r17 , -Y

mov r18 , r17

ldi r19 , $0

rcall find\_rem

cpi r17 , $0

brne skip

add r20 , r18

adc r21 , r19

skip:

dec r16

brne memory\_parse\_loop

end: jmp end

find\_rem:

loop\_start:

sbci r17 , div\_no

cpi r17 , div\_no

brcc loop\_start

ret

|  |  |  |  |
| --- | --- | --- | --- |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 3/C |  |  |

/\*

\* CPE\_301\_Design\_Assignment\_c.asm

\* Created: 2/18/2016 4:32:13 PM

\* Author: blumn

\*/

.nolist

.include "m328pdef.inc"

.equ stacktop = $08DF

.equ div\_no = $3

.list

.dseg

.cseg

.org $000

; init counter

ldi r16 , $19

; init stack pointer

ldi r22 , high(stacktop)

out SPH , r22

ldi r22 , low(stacktop)

out SPL , r22

; init pointer y to RAM end

ldi r28 , low(RAMEND)

ldi r29 , high(RAMEND)

; loop

memory\_parse\_loop:

ld r17 , -Y

mov r18 , r17

ldi r19 , $0

rcall find\_rem

cpi r17 , $0

brne skip

add r20 , r18

adc r21 , r19

skip:

dec r16

brne memory\_parse\_loop

end: jmp end

find\_rem:

loop\_start:

sbci r17 , div\_no

cpi r17 , div\_no

brcc loop\_start

ret

|  |  |  |  |
| --- | --- | --- | --- |
| 4. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 4/D |  |  |

/\*

\* CPE\_301\_Design\_Assignment\_d.asm

\* Created: 2/18/2016 4:40:11 PM

\* Author: blumn

\*/

nolist

.include "m328pdef.inc"

.equ stacktop = $08DF

.equ div\_no1 = $7

.equ div\_no2 = $3

.list

.dseg

.cseg

.org $000

; init counter

ldi r16 , $19

; init stack pointer

ldi r22 , high(stacktop)

out SPH , r22

ldi r22 , low(stacktop)

out SPL , r22

; init pointer y to RAM end

ldi r28 , low(RAMEND)

ldi r29 , high(RAMEND)

; loop

memory\_parse\_loop:

;for divisibility by 7

ld r17 , -Y

mov r18 , r17

ldi r19 , $0

rcall find\_rem7

cpi r17 , $0

brne skip1

add r20 , r18

adc r21 , r19

skip1:

;for divisibility by 3

ld r17 , Y

mov r18 , r17

ldi r19 , $0

rcall find\_rem3

cpi r17 , $0

brne skip2

add r23 , r18

adc r24 , r19

skip2:

dec r16

brne memory\_parse\_loop

brbc $0 , end

andi r22 , $04

mov r7 , r22

end: jmp end

find\_rem7:

loop\_start1:

sbci r17 , div\_no1

cpi r17 , div\_no1

brcc loop\_start1

ret

find\_rem3:

loop\_start2:

sbci r17 , div\_no2

cpi r17 , div\_no2

brcc loop\_start2

ret

|  |  |  |  |
| --- | --- | --- | --- |
| 5. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 5/E |  |  |

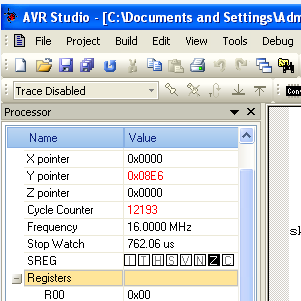
**Formula:**

Clock Cycle= excution\_time x operating\_frequency

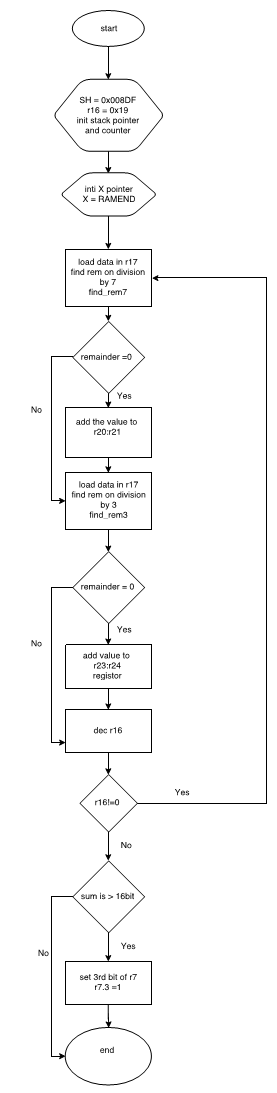
Clock Cycle = 760.06us x 16 MHz = 12193 Clock Cycles

**From Simulation:**

execution\_time = 762.06us

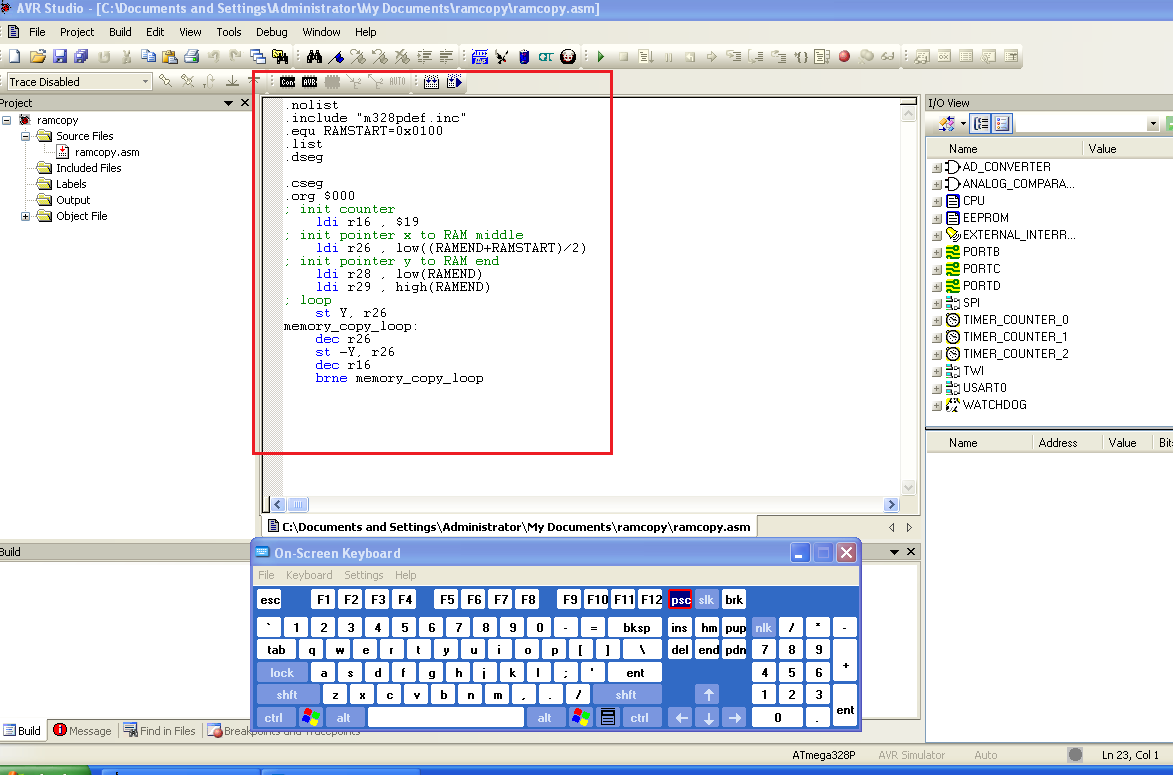
operating\_frequency = 16 MHz  
  


|  |  |  |  |
| --- | --- | --- | --- |
| 6. | Flow Diagram of Final Code |  |  |

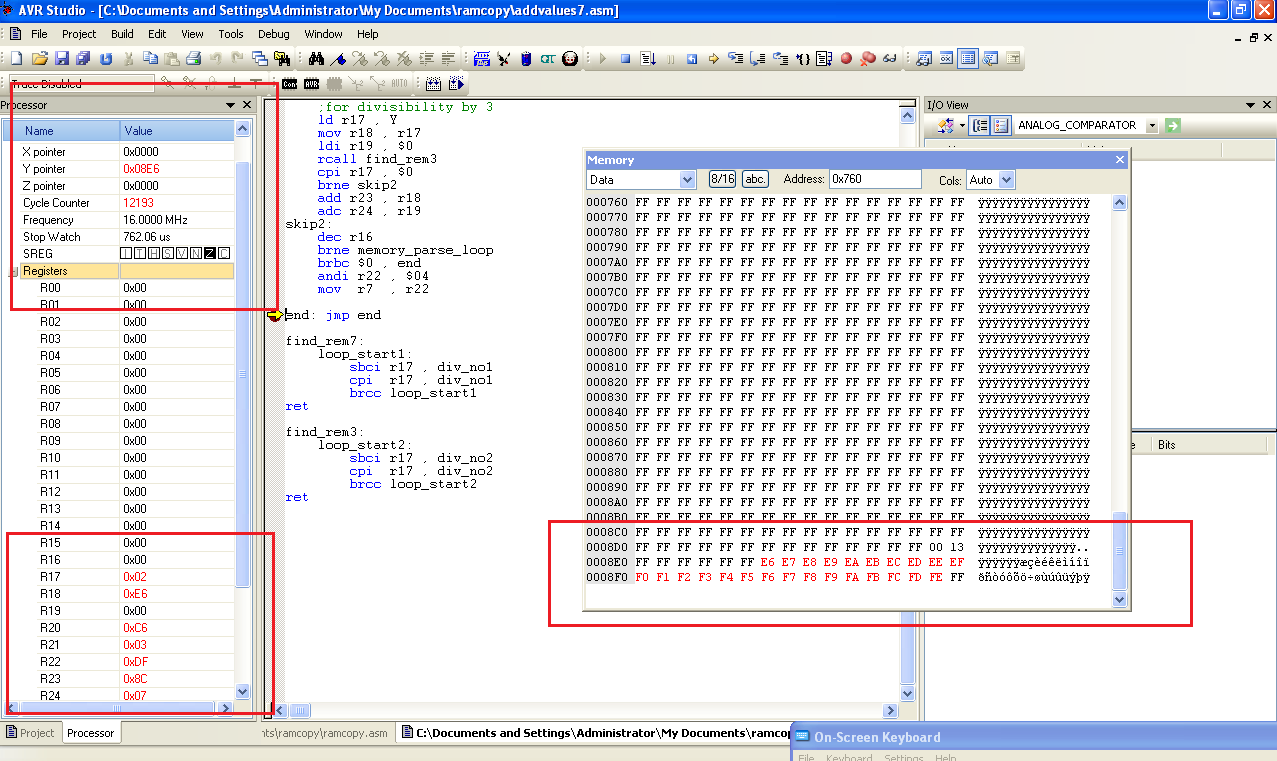


|  |  |  |  |
| --- | --- | --- | --- |
| 7 | SCREENSHOTS OF EACH TASK OUTPUT |  |  |

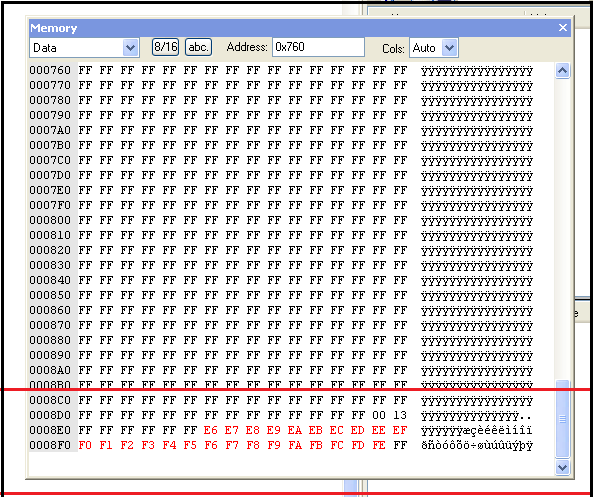
TASK 1/A :

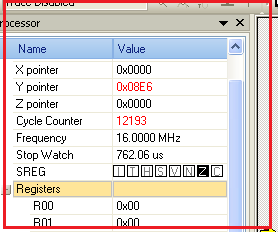


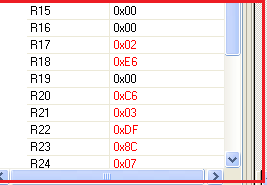
TASK 1/A & 2/B & 3B & 4B



|  |  |  |  |
| --- | --- | --- | --- |
| 8. | SCREENSHOT OF EACH DEMO |  |  |







|  |  |  |  |
| --- | --- | --- | --- |
| 9. | INSTRUCTION DOCUMENT |  |  |
| * Instructions used  1. ldi - load data immediate , loads a constant value to the register 2. out – store data to configuration register 3. ld - load data , it loads data from a register to a location given by pointer register ( X,Y,Z) 4. mov – copy data from one register to the another 5. cpi - compares the value in register to a constant value 6. brne – branch to a location if zero flag is cleared 7. add – add two values inside the given register and save it in frist one 8. adc - add two values with carry 9. rcall – remote call a function 10. brbc – Branch if a given SREG bit is cleared 11. andi – and the values in register with the constant values 12. sbci - sub a constant value from the given register | | | |
| 10. | GOOGLECODE LINK OF THE DA |  |  |
| https://github.com/blumn/DA1 | | | |

**Student Academic Misconduct Policy**

<http://studentconduct.unlv.edu/misconduct/policy.html>

“This assignment submission is my own, original work”.

Noor Blum