Martin Jaime

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 |  |  |
| 1. | INITIAL CODE OF TASK 1/A |  |  |
| 2. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B |  |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 3/C |  |  |
| 4. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 4/D |  |  |
| 5. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 5/E |  |  |
| 6. | SCHEMATICS |  |  |
| 7. | SCREENSHOTS OF EACH TASK OUTPUT |  |  |
| 8. | SCREENSHOT OF EACH DEMO |  |  |
| 9. | VIDEO LINKS OF EACH DEMO |  |  |
| 10. | GOOGLECODE LINK OF THE DA |  |  |
|  |  |  |  |
|  |  |  |  |

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

The only component used in this assignment is AVR Studio 7 simulator. Instead of showing block diagrams, here is the complete code used. In later sections, only part of the following will be shown to point out how a particular task was accomplished.

.def count = r22

.def temp = r25

.def zero = r0

.cseg

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Task a \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

sub zero, zero ;; make zero register.

;; Initialize X and Y pointers to point to ramend / 2

ldi xl, low(ramend / 2)

ldi xh, high(ramend / 2)

movw yl, xl ;; Y = X

mov r16, xl ;; r16 = low(x)

ldi count, 25 ;; count = 25

loop25: ;; Store 25 integers into ramend/2 and up.

st y+, r16 ;; \*y = r16; y++

inc r16 ;; r16++

dec count ;; count--

brne loop25 ;; goto to loop25 if count == 0

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Task b \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

;; reset y register to x

movw yl, xl ;; y = x

ldi count, 25 ;; counter = 25

ldi r17, 7 ;; r17 = 7 // divisor

call Add25 ;; call Add25 routine

mov r20, r10 ;; get return values.

mov r21, r11 ;; r21:r20 <- r11:r10

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Task d \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

;; if r21:r20 is larger than 8 bits, set bit 3 in r7

cp r21, zero ;; check if high byte is 0

breq lessthan8bits7

mov temp, zero

sbr temp, 4

mov r7, temp ;; set bit 3 in r7 is true

lessthan8bits7:

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Task c \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

;; reset y register to x

movw yl, xl ;; y <- x

ldi count, 25 ;; counter = 25

ldi r17, 3 ;; r17 = 3 // divisor

call Add25

mov r23, r10

mov r24, r11 ;; r24:r23 <- r11:r10

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Task d \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

;; if r24:r23 is larger than 8 bits, set bit 3 in r7

cp r24, zero ;; check if high byte is 0

breq lessthan8bits3

mov temp, zero

sbr temp, 4

mov r7, temp ;; set bit 3 in r7 is true

lessthan8bits3:

end:

rjmp end

;; Routine to add 25 integers previously stored divisible by 3.

;; Registers used:

;; r1, r16, r15, r17, r18, r20,

Add25:

clr r10 ;; Clear registers r11:r10

clr r11

add25Loop:

ld r1, y+ ;; get next value

mov r16, r1 ;; store it in r16

call div8u ;; call divide routine

cp r15, zero ;; Check if remainder is 0

brne notDivByR ;;

add r10, r1 ;; Add if remainder is 0

adc r11, zero ;; add values

notDivByR:

dec count ;; loop counter

brne add25Loop

ret ;; return to calling routine

;\*\*\*\* A P P L I C A T I O N N O T E A V R 2 0 0 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* Title: Multiply and Divide Routines

;\* Version: 1.1

;\* Last updated: 97.07.04

;\* Target: AT90Sxxxx (All AVR Devices)

;\*

;\* Support E-mail: avr@atmel.com

;\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* "div8u" - 8/8 Bit Unsigned Division

;\*

;\* This subroutine divides the two register variables "dd8u" (dividend) and

;\* "dv8u" (divisor). The result is placed in "dres8u" and the remainder in

;\* "drem8u".

;\*

;\* Number of words :14

;\* Number of cycles :97

;\* Low registers used :1 (drem8u)

;\* High registers used :3 (dres8u/dd8u,dv8u,dcnt8u)

;\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*\*\*\*\* Subroutine Register Variables

.def drem8u =r15 ;remainder

.def dres8u =r16 ;result

.def dd8u =r16 ;dividend

.def dv8u =r17 ;divisor

.def dcnt8u =r18 ;loop counter

;\*\*\*\*\* Code

div8u: sub drem8u,drem8u ;clear remainder and carry

ldi dcnt8u,9 ;init loop counter

d8u\_1: rol dd8u ;shift left dividend

dec dcnt8u ;decrement counter

brne d8u\_2 ;if done

ret ; return

d8u\_2: rol drem8u ;shift dividend into remainder

sub drem8u,dv8u ;remainder = remainder - divisor

brcc d8u\_3 ;if result negative

add drem8u,dv8u ; restore remainder

clc ; clear carry to be shifted into result

rjmp d8u\_1 ;else

d8u\_3: sec ; set carry to be shifted into result

rjmp d8u\_1

Application note from AVR was used to perform unsigned 8-bit division to obtain the remainder of an operation.

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

Store 25 numbers starting from the RAMEND/2 location. Capture the lower 8bits of the variable/memory location RAM\_MIDDLE = RAMEND/2 address and use them as your values. You can increment or decrement from RAM\_MIDDLE location to get the subsequent 24 numbers. Use the X/Y/Z registers as pointers to fill up 25 numbers starting from location=RAM\_MIDDLE.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Task a \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

sub zero, zero ;; make zero register.

;; Initialize X and Y pointers to point to ramend / 2

ldi xl, low(ramend / 2)

ldi xh, high(ramend / 2)

movw yl, xl ;; Y = X

mov r16, xl ;; r16 = low(x)

ldi count, 25 ;; count = 25

loop25: ;; Store 25 integers into ramend/2 and up.

st y+, r16 ;; \*y = r16; y++

inc r16 ;; r16++

dec count ;; count--

brne loop25 ;; goto to loop25 if count == 0

|  |  |  |  |
| --- | --- | --- | --- |
| 2. | INITIAL CODE OF TASK B |  |  |

Use X/Y/Z register to parse through the 25 numbers and add all numbers divisible by 7 and place the result in R20:21.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Task b \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

;; reset y register to x

movw yl, xl ;; y = x

ldi count, 25 ;; counter = 25

ldi r17, 7 ;; r17 = 7 // divisor

call Add25 ;; call Add25 routine

mov r20, r10 ;; get return values.

mov r21, r11 ;; r21:r20 <- r11:r10

The line

call Add25 ;; call Add25 routine

calls the subroutine Add25, which takes as parameters count, and r17 to add 25 unsigned integers from the location at Y and up. Add25 is implemented as follows.

Add25:

clr r10 ;; Clear registers r11:r10

clr r11

add25Loop:

ld r1, y+ ;; get next value

mov r16, r1 ;; store it in r16

call div8u ;; call divide routine

cp r15, zero ;; Check if remainder is 0

brne notDivByR ;;

add r10, r1 ;; Add if remainder is 0

adc r11, zero ;; add values

notDivByR:

dec count ;; loop counter

brne add25Loop

ret ;; return to calling routine

|  |  |  |  |
| --- | --- | --- | --- |
| 3. | INITIAL CODE OF TASK C |  |  |

Use X/Y/Z register to parse through the 25 numbers and add all numbers divisible by 3 and place the result in R23:24. Parsing of the numbers for task b and c has to be done simultaneously.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Task c \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

;; reset y register to x

movw yl, xl ;; y <- x

ldi count, 25 ;; counter = 25

ldi r17, 3 ;; r17 = 3 // divisor

call Add25

mov r23, r10

mov r24, r11 ;; r24:r23 <- r11:r10

Notice that task C also calls the Add25 subroutine.

|  |  |  |  |
| --- | --- | --- | --- |
| 4. | INITIAL CODE OF TASK D |  |  |

Check and set register R07.3 if the sum is greater than 8-bits.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Task d \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

;; if r24:r23 is larger than 8 bits, set bit 3 in r7

cp r24, zero ;; check if high byte is 0

breq lessthan8bits3

mov temp, zero

sbr temp, 4

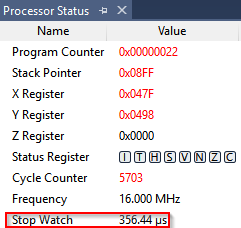
mov r7, temp ;; set bit 3 in r7 is true

Task D is implemented twice. Once after Task B, and again after Task C to check the result of the operations for both tasks. For both tasks, the sum was greater than 8 bits.

|  |  |
| --- | --- |
| For Task B, | For Task C, |
| 0x85 +0x8c + 0x93 = **0x01A4** | 0x81 + 0x84 + 0x87 + 0x8a + 0x8d + 0x90 + 0x93 + 0x96 = **0x045C** |

|  |  |  |  |
| --- | --- | --- | --- |
| 4. | INITIAL CODE OF TASK D |  |  |

Determine the execution time @ 16MHz/#cycles of your algorithm using the simulation.



|  |  |  |  |
| --- | --- | --- | --- |
| 6. | SCHEMATICS |  |  |

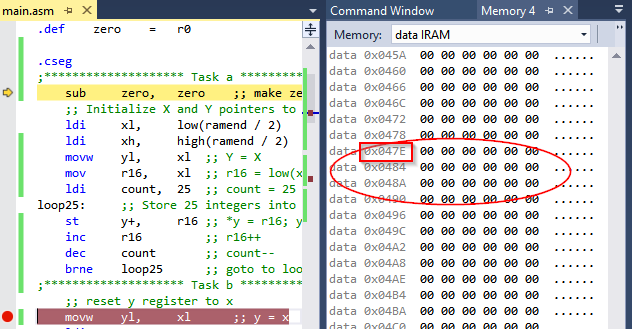
There were no schematics in this assignment.

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

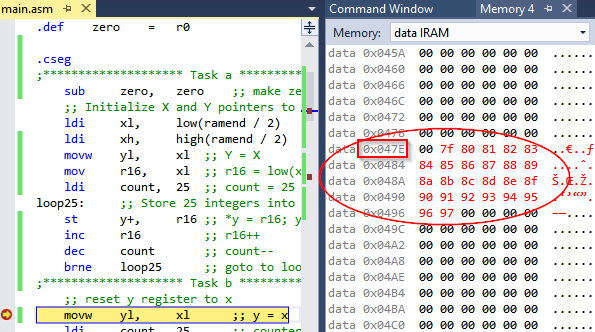
## TASK A:

Starting at RAMEND/2, memory should hold values corresponding to the lower byte of their memory location. If RAMEND = 0x08FF, RAMEND/2 = 0x047F.

Before Task A:



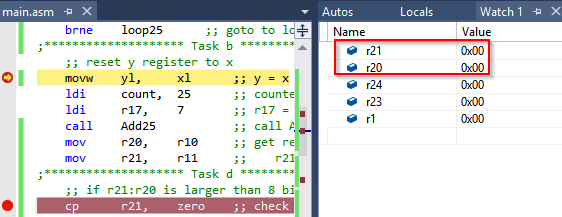
After Task A:



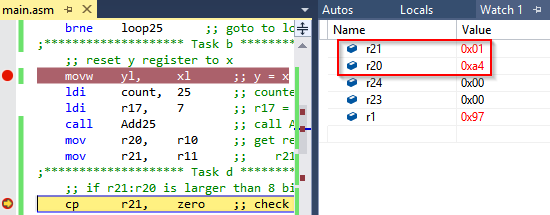
## Task B:

Add all numbers previously stored that are divisible by 7 and place them in r21:r20.

Before Task B:



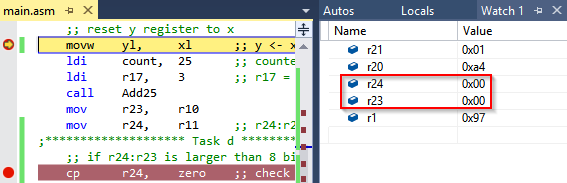
After Task B:



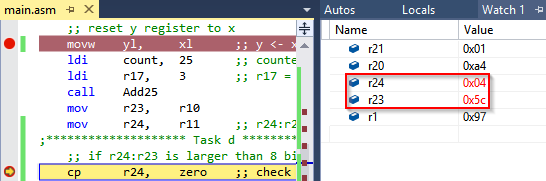
## Task C:

Add all numbers previously stored that are divisible by 3 and place them in r24:r23.

Before Task C:



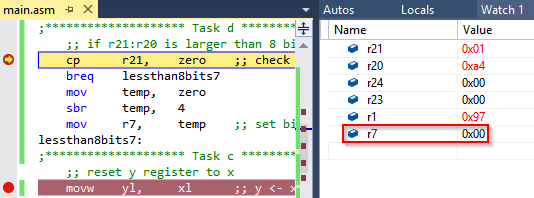
After Task C:



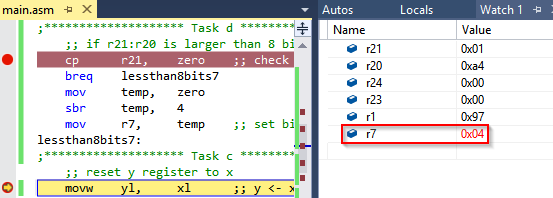
## Task D:

Set bit 3 in register r7 if the result is greater than 8-bits. The following illustrates this task after task b is computed.

Before Task D:



After Task D:



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

See simulation output on previous section.

|  |  |  |  |
| --- | --- | --- | --- |
| 9. | VIDEO LINKS OF EACH DEMO |  |  |
| Videos were not requested. | | | |
| 10. | Github Repository |  |  |
| https://github.com/martinjaime/CpE301\_Assignments2016S.git | | | |

**Student Academic Misconduct Policy**

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

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

Martin Jaime