CPE301 – SPRING 2019

Design Assignment 1

Student Name: Tanner Tindall

Student #: 8000733733

Student Email: tindat1@unlv.nevada.edu

Primary Github address: <https://github.com/TannerTindall51>

Directory: <https://github.com/TannerTindall51/tindalltannerm_submission/tree/master/Design_Assignments/DA1>

Submit the following for all Labs:

1. In the document, for each task submit the modified or included code (only) with highlights and justifications of the modifications. Also, include the comments.
2. Use the previously create a Github repository with a random name (no CPE/301, Lastname, Firstname). Place all labs under the root folder ESD301/DA, sub-folder named LABXX, with one document and one video link file for each lab, place modified asm/c files named as LabXX-TYY.asm/c.
3. If multiple asm/c files or other libraries are used, create a folder LabXX-TYY and place these files inside the folder.
4. The folder should have a) Word document (see template), b) source code file(s) and other include files, c) text file with youtube video links (see template).

1. **COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS**

N/A

1. **INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1/A**

/\*

\* DA1.asm

\*

\* Created: 2/14/2020 6:11:08 PM

\* Author: tindat1

\*/

;

.ORG 0x00 ;Initializing Program Counter to 0

.INCLUDE <m328pdef.inc>

Start:

ldi r16, 0x01 ;Bits 0-7 of the multiplicand (LSB)

ldi r17, 0x02 ;Bits 7-15 of the multiplicand

ldi r18, 0x00 ;Bits 16-23 of the multiplicand

ldi r19, 0x00 ;Bits 24-32 of the multiplicand (MSB)

ldi r20, 0x01 ;Bits 0-7 of the multiplier (LSB)

ldi r21, 0x02 ;Bits 7-15 of the multiplier

ldi r22, 0x00 ;Bits 16-23 of the multiplier

ldi r23, 0x00 ;Bits 24-32 of the multiplier (MSB)

ldi r24, 0x00 ;Bits 0-7 of the result (LSB)

ldi r25, 0x00 ;Bits 7-15 of the result

ldi r26, 0x00 ;Bits 16-23 of the result

ldi r27, 0x00 ;Bits 24-31 of the result

ldi r28, 0x00 ;Bits 32-39 of the result

ldi r29, 0x00 ;Bits 40-47 of the result

ldi r30, 0x00 ;Bits 48-55 of the result

ldi r31, 0x00 ;Bits 56-64 of the result (MSB)

subi r20, 0x01 ;Loops one more time than necassary therefore subtracting by one.

sbci r21, 0x00 ;Carrying over bits from subtraction (if necassary)

sbci r22, 0x00 ;Carrying over bits from subtraction (if necassary)

sbci r23, 0x00 ;Carrying over bits from subtraction (if necassary)

brbs 2, L3 ;If the N flag of the SREG is triggered, code will stop immediately; otherwise continue.

L1:

add r24, r16 ;r16 + r24 into r24

adc r25, r17 ;r17 + r25 + carry into r25

adc r26, r18 ;r18 + r26 + carry into r26

adc r27, r19 ;r19 + r27 + carry into r27

adc r28, r1 ;r28 + r1 + carry into r28 (Note: r1 = 0 therefore only adding carry value from previous operations)

adc r29, r1 ;r29 + r1 + carry into r29 (Note: r1 = 0 therefore only adding carry value from previous operations)

adc r30, r1 ;r30 + r1 + carry into r30 (Note: r1 = 0 therefore only adding carry value from previous operations)

adc r31, r1 ;r31 + r1 + carry into r31 (Note: r1 = 0 therefore only adding carry value from previous operations)

tst r20 ;Utilizing the test function to determine if the value is zero or negative.

brne L2 ;If false, branch to L2. If true, skip next and continue.

tst r21 ;Utilizing the test function to determine if the value is zero or negative.

brne L2 ;If false, branch to L2. If true, skip next and continue.

tst r22 ;Utilizing the test function to determine if the value is zero or negative.

brne L2 ;If false, branch to L2. If true, skip next and continue.

tst r23 ;Utilizing the test function to determine if the value is zero or negative.

brne L2 ;If false, branch to L2. If true, skip next and continue.

jmp L3 ;Once all values of the multiplier reach 0, jumps to L3 to exit program.

L2:

subi r20, 0x01 ;Decrements multiplier by 1

sbci r21, 0x00 ;Carrying over bits from subtraction (if necassary)

sbci r22, 0x00 ;Carrying over bits from subtraction (if necassary)

sbci r23, 0x00 ;Carrying over bits from subtraction (if necassary)

jmp L1 ;Once the miltiplier is decremented by one, the program jumps back to L1 to continue addition operations.

L3:

jmp L3 ;Program completes (infinite loop)

1. **DEVELOPED MODIFIED CODE OF TASK 2/A from TASK 1/A**

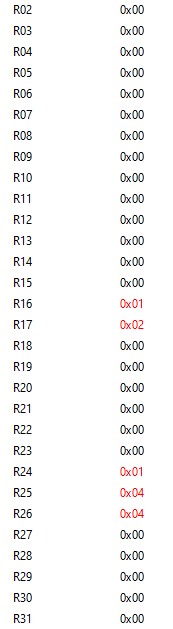
N/A

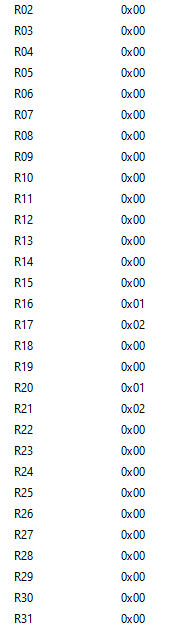
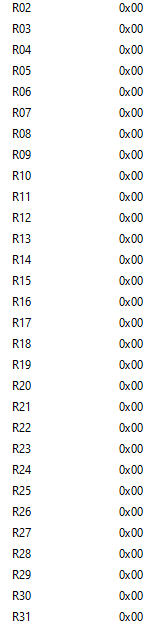
1. **SCHEMATICS**

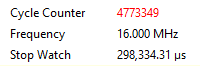
N/A

1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

|  |  |  |
| --- | --- | --- |
| Register values before code runs | Registers holding initialized values | Registers with final values, after calculation |







Note: The final result is stored in registers r24-r26 and confirm the proper result of the two values multiplied.

1. **SCREENSHOT OF EACH DEMO (BOARD SETUP)**

N/A

1. **VIDEO LINKS OF EACH DEMO**

<https://youtu.be/DVxK81nnrfc>

1. **GITHUB LINK OF THIS DA**

<https://github.com/TannerTindall51/tindalltannerm_submission/tree/master/Design_Assignments/DA1>

**Student Academic Misconduct Policy**

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

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

Tanner Tindall