

Design Assignment 1A

Student Name: Jasper Estorco

Student #: 2000836259

Student Email: estorco@unlv.nevada.edu

Primary Github address: https://github.com/toobsock/online_submission

Directory: online_submission/DA/1A

1. DEVELOPED CODE OF TASK 1-2

```
;
; Assign1A.asm
;
; Created: 2/14/2020 11:27:30 AM
; Author : JEstorco
;

.include <m328pbddef.inc>

.ORG 00

; Program does standard multiplication of two 32-bit numbers resulting in a 64-bit sum
; Multiplicand stored in R19-R16 32-bit
; Multiplier stored in R23-R20 32-bit
; SUM R7-0 64-bit

; Operation: 1B5B3731'h (458,962,737) * 1CC355F0'h (482,563,568)
; Result = 312D9B3552A02F0'h (221,478,695,945,765,616)

;Cycles = 513

LOAD_VALUES:
    LDI R25, 33 ; Set Loop Counter to 33
    LDI R20, LOW($000000F0) ; Load LOW byte of Multiplier R20
    LDI R21, HIGH($00005500) ; Load HIGH byte of Multiplier R21
    LDI R22, BYTE3($00C30000) ; Load 3rd byte of Multiplier R22
    LDI R23, BYTE4($1C000000) ; Load 4th byte of Multiplier R23
    LDI R16, LOW($00000031) ; Load LOW byte of Multiplicand R16
    LDI R17, HIGH($00003700) ; Load HIGH byte of Multiplicand R17
```

LDI R18, BYTE3(\$005B0000) ; Load 3rd byte of Multiplicand R18
LDI R19, BYTE4(\$1B000000) ; Load 4th byte of Multiplicand R19
MOV R3, R23 ; Copy 4th byte of Multiplier R23 (Can't use LDI instruction)
MOV R2, R22 ; Copy 3rd byte of Multiplier R22 (Can't use LDI instruction)
MOV R1, R21 ; Copy HIGH byte of Multiplier R21 (Can't use LDI instruction)
MOV R0, R20 ; Copy LOW byte of Multiplier R20 (Can't use LDI instruction)

Iteration:

ROR R3 ; Shift 4th byte (of Multiplier) R23 to right in R3
ROR R2 ; Shift 3rd byte (of Multiplier) R22 to right in R2
ROR R1 ; Shift HIGH byte (of Multiplier) R21 to right in R1
ROR R0 ; Shift LOW byte (of Multiplier) R20 to right in R0
DEC R25; Decrement Loop Counter
BREQ END ; Branch to end of Program if All 32 Bits Completed
BRCC IGNORE ; Branch to skip additon if carry = zero
ADD R4, R16 ; Add LOW byte (of Multiplicand) R16 to R4 and store result to R4
ADC R5, R17 ; Add HIGH bye (of Multiplicand) R17 to R5 and store result to R5
ADC R6, R18 ; Add 3rd byte (of Multiplicand) R18 to R6 and store result to R6
ADC R7, R19 ; Add 4th byte (of Multiplicand) R19 to R7 and store result to R7

IGNORE:

ROR R7 ; Shift 4th byte (of Multiplicand R19) to right in R7
ROR R6 ; Shift 3rd byte (of Multiplicand R18) to right in R6
ROR R5 ; Shift HIGH byte (of Multiplicand R17) to right in R5
ROR R4 ; Shift LOW byte (of Multiplicand R16) to right in R4
RJMP Iteration ; Proceed with remaining Iteration

END:

RJMP END ; Finish program - Infinite Loop

2. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

Beginning of Simulation:

```
; 3232.asm
;
; Created: 2/14/2020 11:27:30 AM
; Author : JEstorco
;

.include <m328pdef.inc>

.ORG 00

; Program does standard multiplication of two 32-bit numbers resulting in a 64-bit sum
; Multiplicand stored in R19-R16 32-bit
; Multiplier stored in R23-R20 32-bit
; SUM R7-0 64-bit

; Operation: 1B5B3731'h (458,962,737) * 1CC355F0'h (482,563,568)
; Result = 312D9B3552A02F0'h (221,478,695,945,765,616)

;Cycles = 513

LOAD_VALUES:
LDI R25, 33 ; Set Loop Counter to 33
LDI R16, LOW($00000031) ; Load LOW byte of Multiplicand R16
LDI R17, HIGH($00003700) ; Load HIGH byte of Multiplicand R17
LDI R18, BYTE3($005B0000) ; Load 3rd byte of Multiplicand R18
LDI R19, BYTE4($1B000000) ; Load 4th byte of Multiplicand R19
LDI R20, LOW($000000F0) ; Load LOW byte of Multiplier R20
LDI R21, HIGH($00005500) ; Load HIGH byte of Multiplier R21
LDI R22, BYTE3($00C30000) ; Load 3rd byte of Multiplier R22
LDI R23, BYTE4($1C000000) ; Load 4th byte of Multiplier R23
MOV R0, R20 ; Copy LOW byte of Multiplier R20 to
MOV R1, R21 ; Copy HIGH byte of Multiplier R21
MOV R2, R22 ; Copy 3rd byte of Multiplier R22
MOV R3, R23 ; Copy 4th byte of Multiplier R23

Iteration:
```

Register Window:

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	= 0x31
R17	= 0x37
R18	= 0x5B
R19	= 0x1B
R20	= 0xF0
R21	= 0x55
R22	= 0xC3
R23	= 0x1C
R24	= 0x00
R25	= 0x21
R26	= 0x00
R27	= 0x00
R28	= 0x00
R29	= 0x00
R30	= 0x00
R31	= 0x00

Multiplicand and Multiplier Loaded into the Registers

End of Simulation:

The screenshot displays a microcontroller simulator interface. The main window shows assembly code with the following instructions:

```
ROR R2 ; Shift 4th byte (of Multiplier) R23 to right in R2
ROR R2 ; Shift 3rd byte (of Multiplier) R22 to right in R2
ROR R1 ; Shift HIGH byte (of Multiplier) R21 to right in R1
ROR R0 ; Shift LOW byte (of Multiplier) R20 to right in R0
DEC R25 ; Decrement Loop Counter
BRCC END ; Branch to end of Program if All 32 Bits Completed
BRCC IGNORE ; Branch to skip additon if carry = zero
ADD R4, R16 ; Add LOW byte of Multiplicand R16 to R4 and store result to R4
ADC R5, R17 ; Add HIGH bye of Multiplicand R17 to R5 and store result to R5
ADC R6, R18 ; Add 3rd byte of Multipicand R18 to R6 and store result to R6
ADC R7, R19 ; Add 4th byte of Multipicand R19 to R7 and store result to R7
```

Below the code, the 'IGNORE:' section contains:

```
ROR R7 ; Shift 4th byte (of Multiplicand R19) to right in R7
ROR R6 ; Shift 3rd byte (of Multiplicand R18) to right in R6
ROR R5 ; Shift HIGH byte (of Multiplicand R17) to right in R5
ROR R4 ; Shift LOW byte (of Multiplicand R16) to right in R4
RJMP Iteration ; Proceed with remaining Iteration
```

The 'END:' section contains:

```
RJMP END ; Finish program - Infinite Loop
```

On the right side, a list of register values is shown, with registers R00 through R07 highlighted in a red box:

- R00 = 0xF0
- R01 = 0x02
- R02 = 0x2A
- R03 = 0x55
- R04 = 0xB3
- R05 = 0xD9
- R06 = 0x12
- R07 = 0x03
- R08 = 0x00
- R09 = 0x00
- R10 = 0x00
- R11 = 0x00
- R12 = 0x00
- R13 = 0x00
- R14 = 0x00
- R15 = 0x00
- R16 = 0x31
- R17 = 0x37
- R18 = 0x5B
- R19 = 0x1B
- R20 = 0xF0
- R21 = 0x55
- R22 = 0xC3
- R23 = 0x1C
- R24 = 0x00
- R25 = 0x00
- R26 = 0x00
- R27 = 0x00
- R28 = 0x00
- R29 = 0x00
- R30 = 0x00
- R31 = 0x00

A red oval labeled '64-Bit Sum Stored' points to the registers R00 through R07.

The 'Processor Status' window is open, showing the following values:

Name	Value
Program Counter	0x0000001D
Stack Pointer	0x08FF
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	1 1 1 1 1 1 1 1
Cycle Counter	513
Frequency	1.000 MHz
Stop Watch	513.00 µs

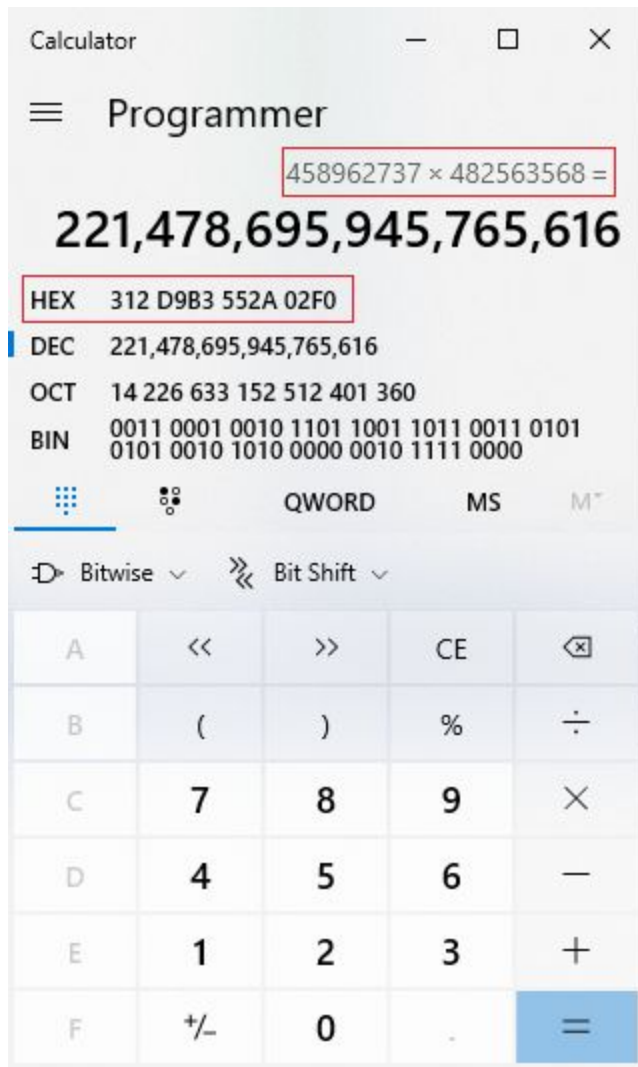
A red oval labeled 'Cycle count of 513' points to the 'Cycle Counter' value in the Processor Status window.

The 'Registers' section of the Processor Status window shows the following values:

Register	Value
R00	0xF0
R01	0x02
R02	0x2A
R03	0x55


Multiplication with Iterations completed, sum stored in 8 Registers for a 64-bit result. Processor Status window from debugger measures 513 cycles.

Calculation:



Execution Time:

16MHz clock / 513 cycles = 31.2 μ s

Program Counter	0x0000001D
Stack Pointer	0x08FF
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	
Cycle Counter	513
Frequency	16.000 MHz
Stop Watch	32.06 μ s

3. GITHUB LINK OF THIS DA

Student Academic Misconduct Policy

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

"This assignment submission is my own, original work".

Jasper Estorco