CPE301 – SPRING 2019

Design Assignment 1A

Student Name: Dan Ray Dimapilis
Student #: 5002763761
Student Email: dimapd1@unlv.nevada.edu
Primary Github address: https://github.com/dimapd1/ASN_subs
Directory: dimapd1/ASN_subs

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

NA

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

```
;AssemblerApplication1.asm
;manually mul 32b by 32b to get 64b product while not using mul instruction
; Created: 2/13/2020 6:06:59 AM
; Author : dimapd1
;r[19,18,17,16] - multiplicand
;r[23,22,21,20] - multiplier
;r[Z, Y, X, 25 24] -product
;Z(H/L) = r31/30        Y=29/28                X=27/26

 .DEF  ANS1 = r24        ;64'b product
 .DEF  ANS2 = r25        ;
 .DEF  ANS3 = r26        ;
 .DEF  ANS4 = r27        ;
 .DEF  ANS5 = r28        ;
 .DEF  ANS6 = r29        ;
 .DEF  ANS7 = r30        ;
 .DEF  ANS8 = r31        ;
 .DEF    A1 = R16        ;32'b Multiplicand
 .DEF    A2 = R17        ;
 .DEF    A3 = R18        ;
 .DEF    A4 = R19        ;
 .DEF    B1 = R20        ;32'b Multiplier
 .DEF    B2 = R21        ;
 .DEF    B3 = R22        ;
 .DEF    B4 = R23        ;
   ;   Ctr = R0        ;Lp Counter

initializations:
                ldi r16, 33
                mov r0, r16                                    ;Set Loop Counter to 33
```

```
    LDI   A1,  LOW($190f19a0)        ;load imm, low byte of ?420 420 000? decimal
    LDI   A2,BYTE2($190f19a0)        ;load imm, 2nd byte of ?420 420 000? decimal
    LDI   A3,BYTE3($190f19a0)        ;load imm, 3rd byte of ?420 420 000? decimal
    LDI   A4,BYTE4($190f19a0)        ;load imm, 4th byte of ?420 420 000? decimal
    LDI   B1,  LOW($00000208)        ;load imm, low byte of 520 decimal
    LDI   B2,BYTE2($00000208)        ;load imm, second byte of 520 decimal
    LDI   B3,BYTE3($00000208)        ;load imm, third byte of 520 decimal
    LDI   B4,BYTE4($00000208)        ;load imm, fourth byte of 520 decimal

calculations:
    CLR   ANS1       ;Initialize Answer to zero
    CLR   ANS2       ;
    CLR   ANS3       ;
    CLR   ANS4       ;
    CLR   ANS5       ;
    CLR   ANS6       ;
    CLR   ANS7       ;
    SUB   ANS8,ANS8    ;Clear ANS8 and Carry Flag
    MOV   ANS1,B1      ;Copy Multiplier to Answer
    MOV   ANS2,B2      ;
    MOV   ANS3,B3      ;
    MOV   ANS4,B4      ;

LOOP:
    ROR   ANS4       ;Shift Multiplier to right
    ROR   ANS3       ;
    ROR   ANS2       ;
    ROR   ANS1       ;
    DEC   r0        ;Decrement Loop Counter
    BREQ  DONE        ;Check if all bits processed
    BRCC  SKIP_add     ;If Carry Clear skip addition
    ADD   ANS5,A1      ;Add Multipicand into Answer
    ADC   ANS6,A2      ;
    ADC   ANS7,A3      ;
    ADC   ANS8,A4      ;
SKIP_add:
    ROR   ANS8       ;Shift high bytes of Answer
    ROR   ANS7       ;
    ROR   ANS6       ;
    ROR   ANS5       ;
    RJMP LOOP
DONE:  jmp DONE
```

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

NA

4. SCHEMATICS

NA

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

```
initializations:
        ldi r16, 33
        mov r0, r16
        LDI   A1,  LOW($190f19a0)   ;load imm, low byte of 420 420 000 decimal
        LDI   A2,BYTE2($190f19a0)   ;load imm, 2nd byte of 420 420 000 decimal
        LDI   A3,BYTE3($190f19a0)   ;load imm, 3rd byte of 420 420 000 decimal
        LDI   A4,BYTE4($190f19a0)   ;load imm, 4th byte of 420 420 000 decimal
        LDI   B1,  LOW($00000208)   ;load imm, low byte of 520 decimal
        LDI   B2,BYTE2($00000208)   ;load imm, second byte of 520 decimal
        LDI   B3,BYTE3($00000208)   ;load imm, third byte of 520 decimal
        LDI   B4,BYTE4($00000208)   ;load imm, fourth byte of 520 decimal

calculations:
        CLR   ANS1          ;Initialize Answer to zero
        CLR   ANS2          ;
        CLR   ANS3          ;
        CLR   ANS4          ;
```

Watch 2

Name	Value	Type
R16	0xa0	byte{registers}@R16
R17	0x19	byte{registers}@R17
R18	0x0f	byte{registers}@R18
R19	0x19	byte{registers}@R19
R20	0x08	byte{registers}@R20
R21	0x02	byte{registers}@R21
R22	0x00	byte{registers}@R22
R23	0x00	byte{registers}@R23
r0	0x21	byte{registers}@R00

Watch 1

Name	Value	Type
r24	0x00	byte{registers}@R24
r25	0x00	byte{registers}@R25
r26	0x00	byte{registers}@R26
r27	0x00	byte{registers}@R27
r28	0x00	byte{registers}@R28
r29	0x00	byte{registers}@R29
r30	0x00	byte{registers}@R30
r31	0x00	byte{registers}@R31

```
        ldi r16, 33
        mov r0, r16
        LDI   A1,  LOW($190f19a0)   ;load imm, low byte of 420 420 000 decimal
        LDI   A2,BYTE2($190f19a0)   ;load imm, 2nd byte of 420 420 000 decimal
        LDI   A3,BYTE3($190f19a0)   ;load imm, 3rd byte of 420 420 000 decimal
        LDI   A4,BYTE4($190f19a0)   ;load imm, 4th byte of 420 420 000 decimal
        LDI   B1,  LOW($00000208)   ;load imm, low byte of 520 decimal
        LDI   B2,BYTE2($00000208)   ;load imm, second byte of 520 decimal
        LDI   B3,BYTE3($00000208)   ;load imm, third byte of 520 decimal
        LDI   B4,BYTE4($00000208)   ;load imm, fourth byte of 520 decimal

calculations:
        CLR   ANS1          ;Initialize Answer to zero
        CLR   ANS2          ;
        CLR   ANS3          ;
        CLR   ANS4          ;
        CLR   ANS5          ;
        CLR   ANS6          ;
        CLR   ANS7          ;
        SUB   ANS8,ANS8     ;Clear ANS8 and Carry Flag
        MOV   ANS1,B1       ;Copy Multiplier to Answer
        MOV   ANS2,B2       ;
        MOV   ANS3,B3       ;
        MOV   ANS4,B4       ;
       ; LDI   Ctr,33        ;Set Loop Counter to 33
LOOP:
```

Calculator

Programmer

420420000 × 520 =

218,618,400,000

HEX 32 E6AC 0D00

DEC 218,618,400,000

OCT 3 134 653 006 400

BIN 0011 0010 1110 0110 1010 1100 0000 1101 0000 0000

QWORD MS

Bitwise Bit Shift

A << >> CE

B ( ) % ÷

C 7 8 9 ×

D 4 5 6 −

E 1 2 3 +

F +/- 0 . =

R28 = 0x32

Value	Type
0xa0	byte{registers}@R16
0x19	byte{registers}@R17
0x0f	byte{registers}@R18
0x19	byte{registers}@R19
0x08	byte{registers}@R20
0x02	byte{registers}@R21
0x00	byte{registers}@R22
0x00	byte{registers}@R23
0x00	byte{registers}@R00

Watch 1

Name	Value	Type
r24	0x00	byte{registers}@R24
r25	0x0d	byte{registers}@R25
r26	0xac	byte{registers}@R26
r27	0xe6	byte{registers}@R27
r28	0x32	byte{registers}@R28
r29	0x00	byte{registers}@R29
r30	0x00	byte{registers}@R30
r31	0x00	byte{registers}@R31

Memory 4

Memory: prog FL

```
prog 0x0000
prog 0x000F
prog 0x001E
prog 0x002D
prog 0x003C
prog 0x004B
prog 0x005A
```

Watch 1

Name	Value	Type
r24	0x00	byte{registers}@R24
r25	0x0d	byte{registers}@R25
r26	0xac	byte{registers}@R26
r27	0xe6	byte{registers}@R27
r28	0x32	byte{registers}@R28
r29	0x00	byte{registers}@R29
r30	0x00	byte{registers}@R30
r31	0x00	byte{registers}@R31

Processor Status

Name	Value
Program Counter	0x00000026
Stack Pointer	0x08FF
X Register	0xE6AC
Y Register	0x0032
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	483
Frequency	1.000 MHz
Stop Watch	483.00 µs
Registers	
R00	0x00
R01	0x00
R02	0x00

CYCLE COUNT = 483 FOR THE ENTIRE PROCESS OF THE CODE/DA1

6. SCREENSHOT OF EACH DEMO (BOARD SETUP)

NA

7. VIDEO LINKS OF EACH DEMO

NA

8. GITHUB LINK OF THIS DA

Student Academic Misconduct Policy
http://studentconduct.unlv.edu/misconduct/policy.html

"*This assignment submission is my own, original work*".
NAME OF THE STUDENT