Sittthiphol Yuwanaboon

John Oneal

CS 130

4 May 2021

LAB 6

Part A

6502, Multiplying with the add and shift algorithm

1. Test with 0x5 & 0x20 and print screen at the end. Label it PtA1. What was the answer? 0xA0

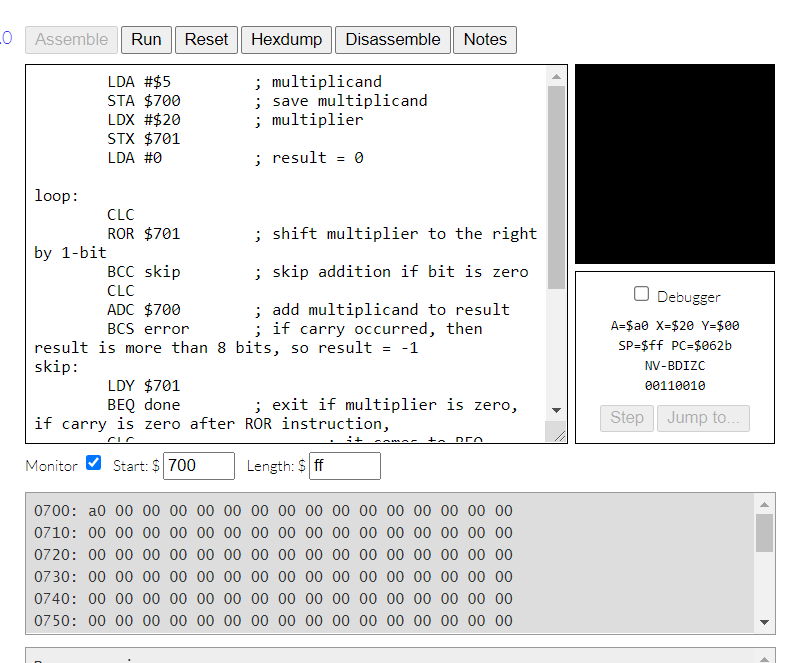


Figure. PtA1

1. Test with 0x3 & 0x20 and print screen at the end. Label it PtA2. What was the answer? 0x60

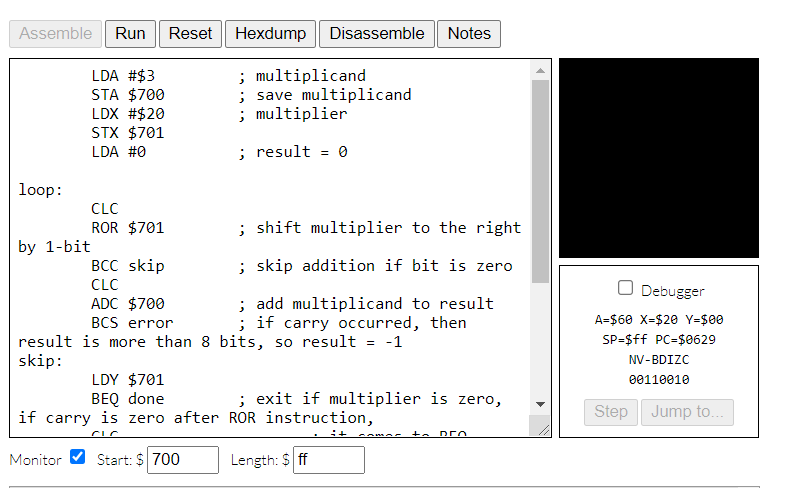


Figure. PtA2

1. Test with 0x8 & 0x31 and print screen at the end. Label it PtA3. What was the answer? 0xFF

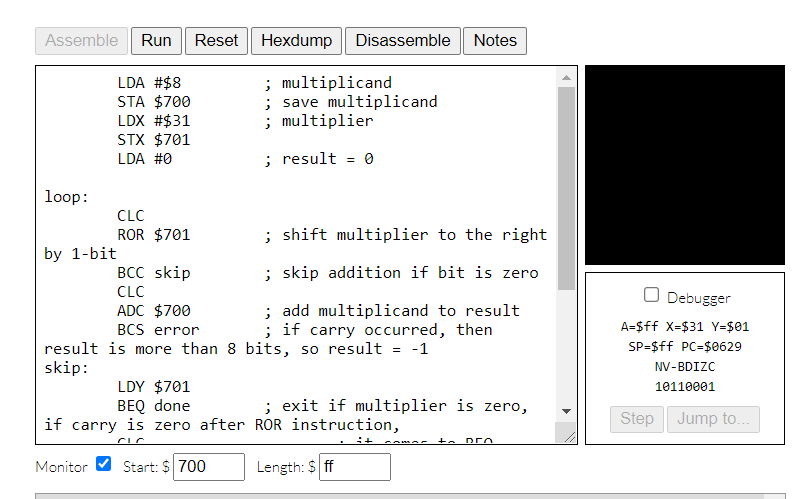


Figure PtA3

Code:

LDA #$5 ; multiplicand

STA $700 ; save multiplicand

LDX #$20 ; multiplier

STX $701

LDA #0 ; result = 0

loop:

CLC

ROR $701 ; shift multiplier to the right by 1-bit

BCC skip ; skip addition if bit is zero

CLC

ADC $700 ; add multiplicand to result

BCS error ; if carry occured, then result is more than 8 bits, so result = -1

skip:

LDY $701

BEQ done ; exit if multiplier is zero, if carry is zero after ROR instruction,

CLC ; it comes to BEQ instruction, where it checks zero flag of the multiplier

ROL $700 ; shift multiplicand to the left by 1-bit

BCS error ; more than 8-bit

JMP loop

error:

LDA #$FF ; result = -1

done:

BRK

Part B

ARM, Multiplying with the add and shift algorithm

1. Use 123657 and 8. Label the screenshot PtB1

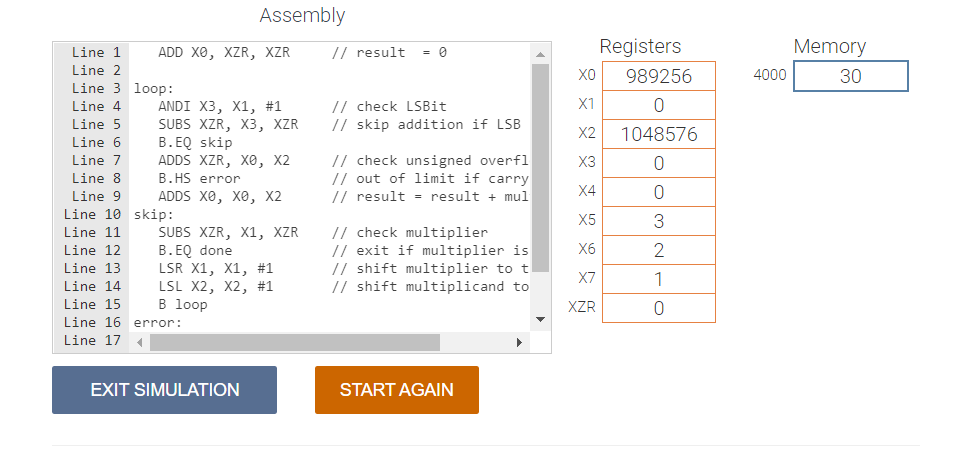


Figure PtB1

1. Use 9,000,000,657,839,333,222 and 2. Label the screenshot PtB2

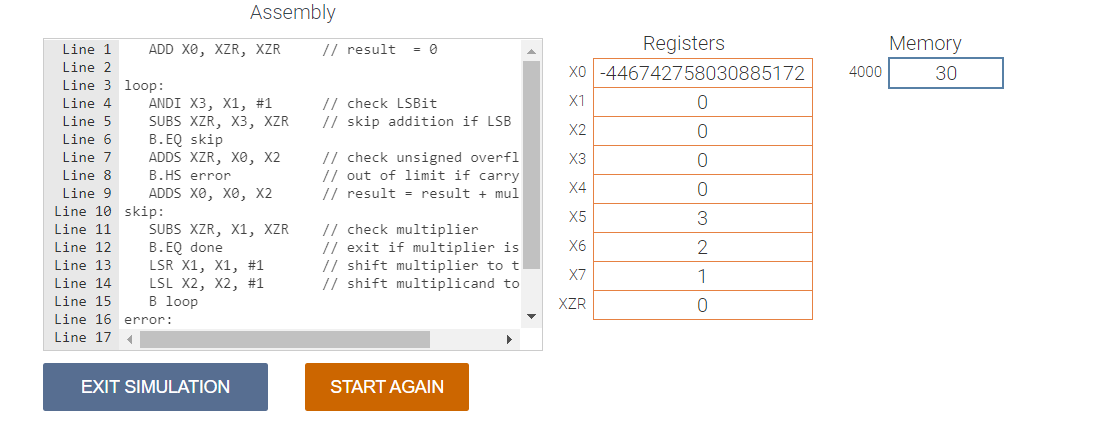


Figure PtB2

1. Use 123,700,00 and 11,777. Label the screenshot PtB3.

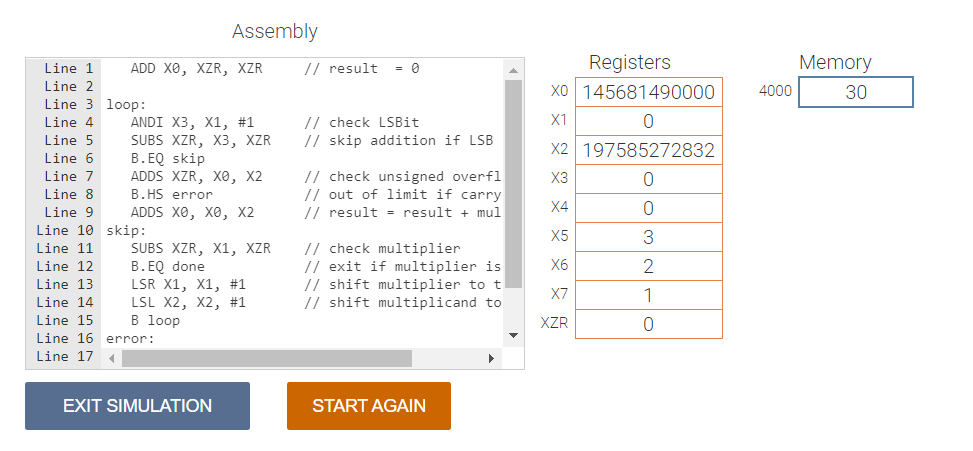


Figure PtB3

Code:

ADD X0, XZR, XZR // result = 0

loop:

ANDI X3, X1, #1 // check LSBit

SUBS XZR, X3, XZR // skip addition if LSB is zero

B.EQ skip

ADDS XZR, X0, X2 // check unsigned overflow occurs

B.HS error // out of limit if carry set

ADDS X0, X0, X2 // result = result + multiplicand

skip:

SUBS XZR, X1, XZR // check multiplier

B.EQ done // exit if multiplier is zero

LSR X1, X1, #1 // shift multiplier to the right by 1-bit

LSL X2, X2, #1 // shift multiplicand to the left by 1-bit

B loop

error:

SUBI X0, XZR, #1 // result = -1

done:

Part C

Multiplying two eight-bit numbers with the add and shift algorithm. The product should be 16 bits.

1. Test with 0xff & 0xff and print screen at the end. Label it PtC1. What was the answer? 0xFF E1

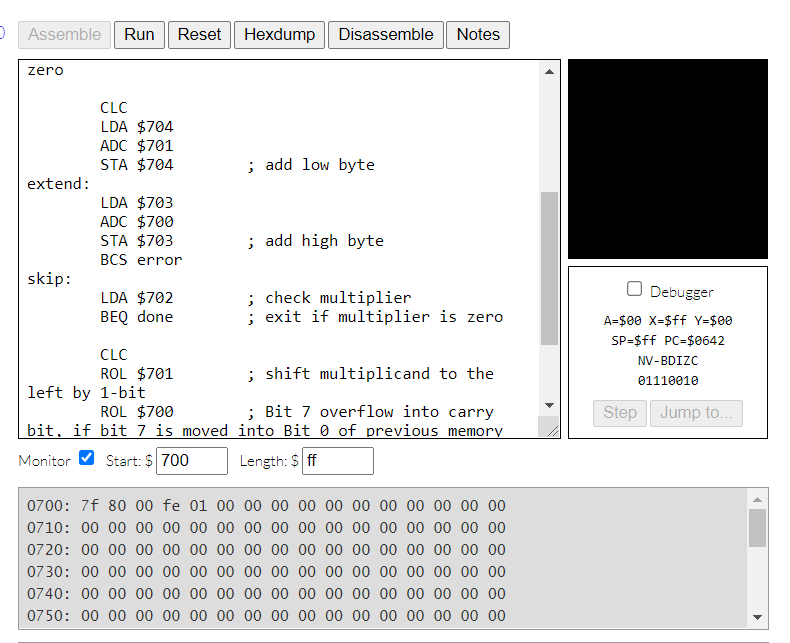


Figure PtC1

1. Test with 0x78 & 0x39 and print screen at the end. Label it PtC2. What was the answer? 0x1A B8

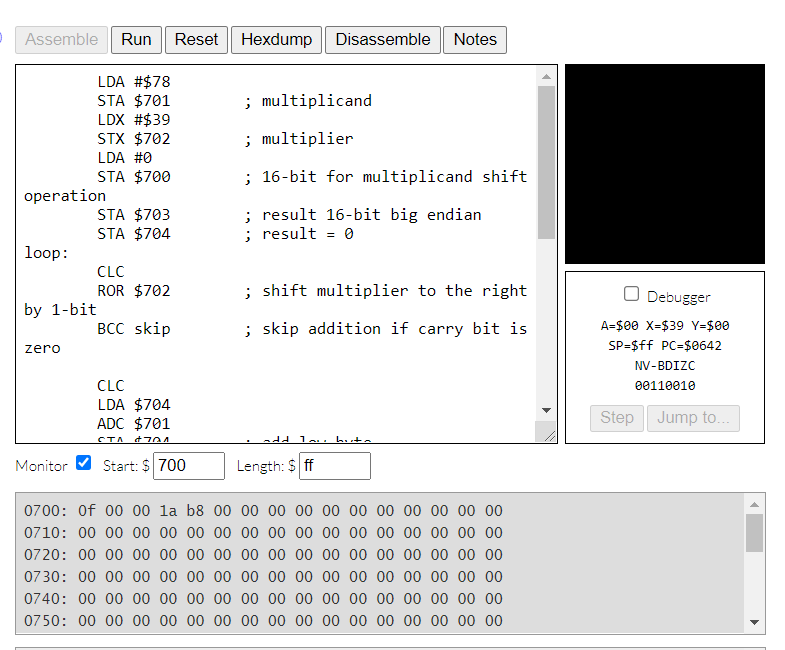


Figure PtC2

1. Test with 0x14 & 0x5 and print screen at the end. Label it PtC3. What was the answer? 0x00 64

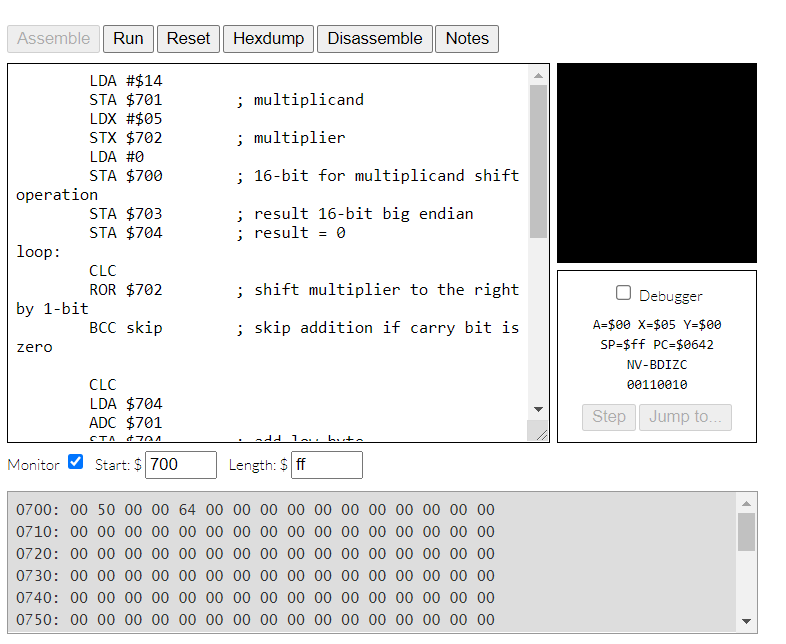


Figure PtC3

Code:

LDA #$14

STA $701 ; multiplicand

LDX #$5

STX $702 ; multiplier

LDA #0

STA $700 ; 16-bit for multiplicand shift operation

STA $703 ; result 16-bit big endian

STA $704 ; result = 0

loop:

CLC

ROR $702 ; shift multiplier to the right by 1-bit

BCC skip ; skip addition if carry bit is zero

CLC

LDA $704

ADC $701

STA $704 ; add low byte

extend:

LDA $703

ADC $700

STA $703 ; add high byte

BCS error

skip:

LDA $702 ; check multiplier

BEQ done ; exit if multiplier is zero

CLC

ROL $701 ; shift multiplicand to the left by 1-bit

ROL $700 ; Bit 7 overflow into carry bit, if bit 7 is moved into Bit 0 of previous memory

BCS error

JMP loop

error:

done:

BRK