­­­­­Sitthiphol Yuwanaboon

John O’Neal

CS 130

21 May 2021

Do all of the PA's in section 2.8 (15 pts.).

NOTE: I'll download these from zybooks for grading.

Write a small program in LegV8 emulator with the following spec's:

Use X19 to input the ASCII code for A, S, R, and L. A is for Add, S is for Subtract,, R is for Right shift, and L is for Left shift.

The main program will take the input in X19 and use it to determine which one of four procedures to call. Each procedure will accept two parameters in registers X0 and X1.

For the add routine, the two parameters are two numbers to be added; the result will be in X3.

For the Subtract routine, the two parameters will be used as (X0)-(X1); the result will be in X3.

For the shift routines, X0 will have the data to be shifted, an X1 will have the number of bits; move the result to X2.

CODE SECTION: Paste your code here.

main:

SUBIS XZR, X19, #65 // check if byte is 'A'

B.NE subt // check next option if not 'A'

BL addition // else call addition procedure

B exit // terminate the code

subt:

SUBIS XZR, X19, #83 // check if byte is 'S'

B.NE lshift // check next option if not

BL subtraction // else call subtraction procedure

B exit // terminate the code

lshift:

SUBIS XZR, X19, #76 // check if byte is 'L'

B.NE rshift // check next option if not

BL leftShift // else call left shift procedure

B exit // terminate the code

rshift:

SUBIS XZR, X19, #82 // check if byte is 'R'

B.NE exit // terminate code if not

BL rightShift // else call right shift procedure

B exit // terminate the code

addition:

ADD X3, X0, X1 // X3 = X0 + X1

BR X30 // return from procedure

subtraction:

SUB X3, X0, X1 // X3 = X0 - X1

BR X30 // return from procedure

leftShift:

LSL X0, X0, #1 // left shift once

SUBI X1, X1, #1 // loop till number of bits shift left

SUBS XZR, X1, XZR // check if counter reached zero

B.GT leftShift // if not loop

ORR X2, XZR, X0 // copy the result

BR X30 // return from procedure

rightShift:

LSR X0, X0, #1 // right shift once

SUBI X1, X1, #1 // loop till number of bits shift right

SUBS XZR, X1, XZR // check if counter reached zero

B.GT rightShift // if not loop

ORR X2, XZR, X0 // copy the result

BR X30 // return from procedure

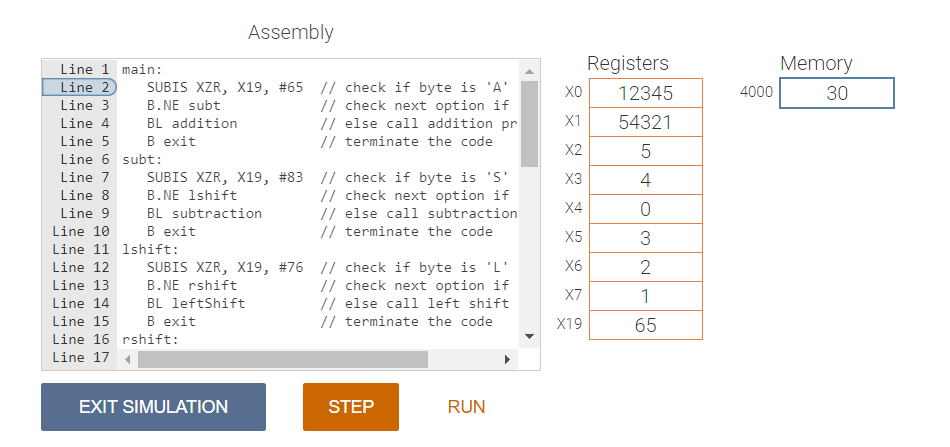
exit:

Picture section:

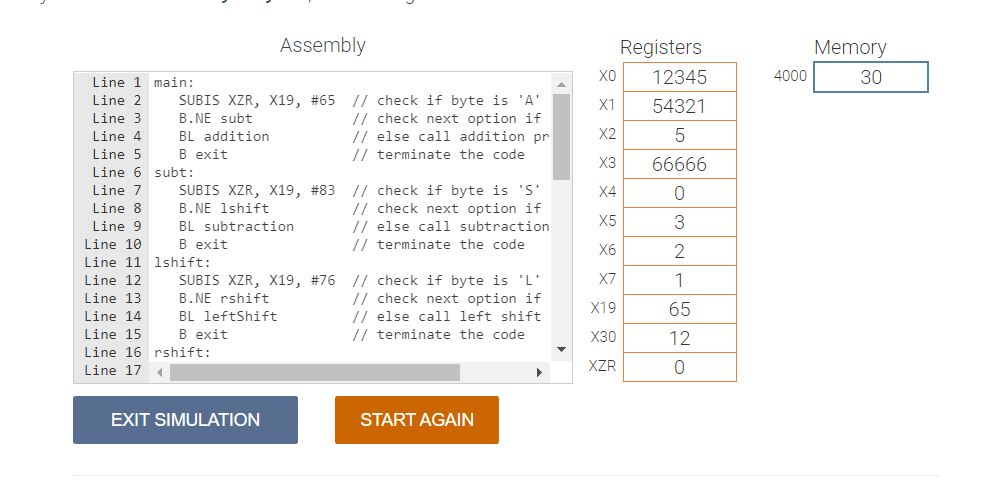
Paste the screenshots using the ADD subroutine and the LftShft routine.

Add 12345 and 54321.

Initial Conditions

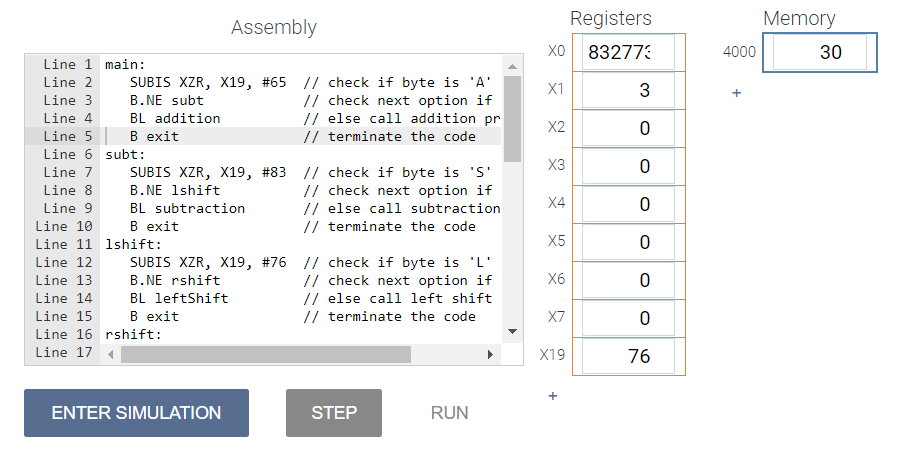


After Simulation

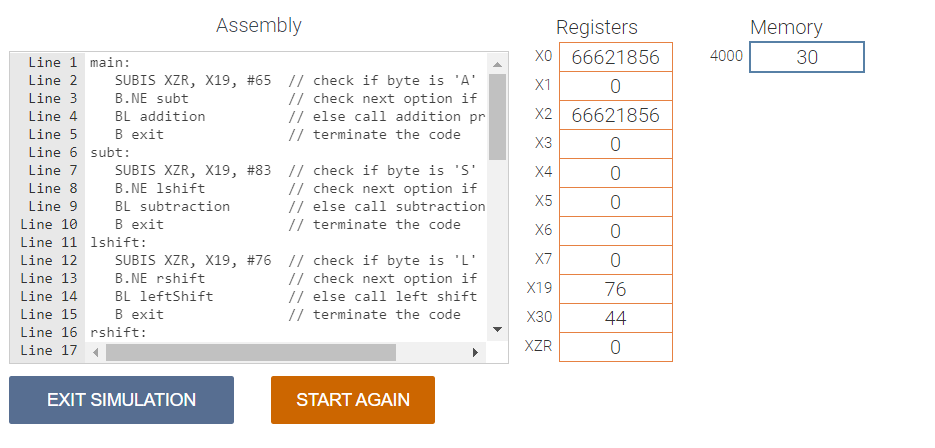


Shift 0x7F1234 three bits for the Laftshift.

Initial Condition



After Simulation



Do the same thing in 6502 assembly and architecture as you did in #2 for the LegV8.

REF: Example of callee cleaning the stack (6502), from Course Files

CODE SECTION: Paste your code here.

main:

LDA #$3A ; first parameter

PHA ; push onto stack

LDA #$0E ; second parameter

PHA ; push onto stack

LDA #65 ; operations 'A'(65), 'S'(83), 'L'(76), 'R'(82)

CMP #65 ; check if 'A'

BNE subt ; if not check for subtraction

JSR addition ; call addition procedure

JMP exit ; terminate the code

subt:

CMP #83 ; check if 'S'

BNE lshift ; if not check for shift operation

JSR subtraction ; call subtraction procedure

JMP exit ; terminate the code

lshift:

CMP #76 ; check if 'L'

BNE rshift ; if not check for right shift

JSR left\_shift ; call left shift procedure

JMP exit ; terminate the code

rshift:

CMP #82 ; check if 'R'

BNE exit ; terminate the code

JSR right\_shift ; call right shift procedure

exit:

BRK ; terminate the code

addition:

TSX ; X = SP

CLC ; clear carry

LDA $104, X ; get first parameter

ADC $103, X ; add to second parameter

RTS ; return from procedure

subtraction:

TSX ; X = SP

SEC ; set carry

LDA $104, X ; get first parameter

SBC $103, X ; subtract from second parameter

RTS ; return from procedure

left\_shift:

TSX ; X = SP

CLC ; clear carry

LDA $104, X ; load first parameter

LDY $103, X ; load second parameter

lshift\_loop:

ROL ; shift left

DEY ; shift till the counter becomes zero

BNE lshift\_loop

RTS ; return from procedure

right\_shift:

TSX ; X = SP

CLC ; clear carry

LDA $104, X ; load first parameter

LDY $103, X ; load second parameter

rshift\_loop:

ROR ; shift right

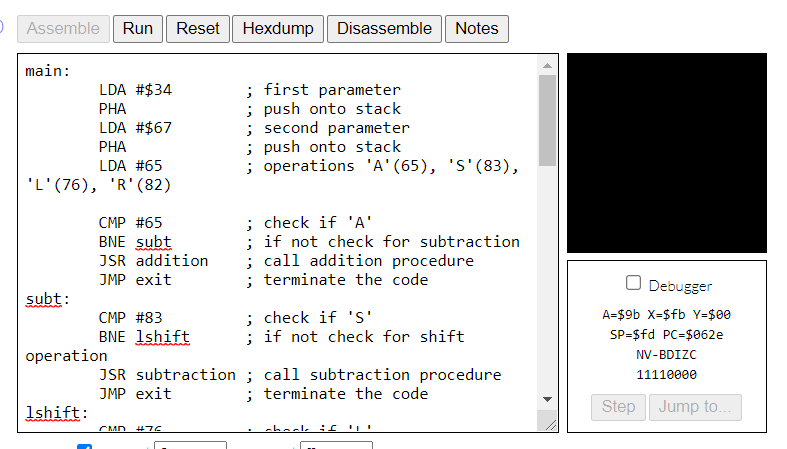
DEY ; shift till the counter becomes zero

BNE rshift\_loop

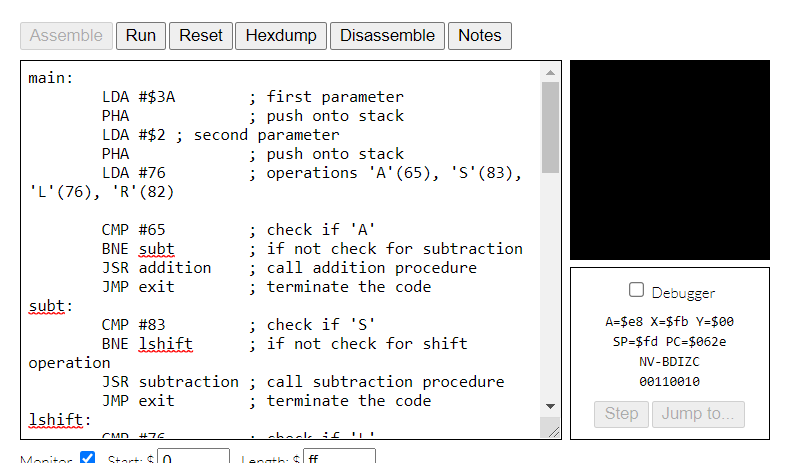
RTS ; return from procedure

Picture Section:

For the Add, add 34h and 67h.



Shift 3Ah two bits.



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_