**Sitthiphol Yuwanaboon**

**John O’neal**

**CS 130**

**May 11 2021**

**2.18** Assume the following register contents: 

X10 = 0x00000000AAAAAAAA, X11 = 0x1234567812345678

**2.18.1** [5] <COD §2.6> For the register values shown above, what is the value of X12 for the following sequence of instructions? 

LSL X12, X10, #4

ORR X12, X12, X11

**2.18.2** [5] <COD §2.6> For the register values shown above, what is the value of X12 for the following sequence of instructions? 

LSL X12, X11, #4

**2.18.3** [5] <COD §2.6> For the register values shown above, what is the value of X12 for the following sequence of instructions? 

LSR X12, X10, #3

ANDI X12, X12, 0xFEF

X10 = 00000000 00000000 00000000 00000000 10101010 10101010 10101010 10101010

X11 = 00010010 00110100 01010110 01111000 00010010 00110100 01010110 01111000

***Ans 2.18.1)*** LSL X12, X10, #4 means X10 is shifted to left by 4 digits and stored in X12, i.e ,

X12 = 00000000 00000000 00000000 00001010 10101010 10101010 10101010 10100000

ORR X12, X12, X11 means (X12) OR (X11) is stored in X12, i.e ,

X12 = 00000000 00000000 00000000 00001010 10101010 10101010 10101010 10100000

**OR**

X11 = 00010010 00110100 01010110 01111000 00010010 00110100 01010110 01111000

gives,

X12 = 00010010 00110100 01010110 01111010 10111010 10111110 11111110 11111000

which can also be written as,

***X12 = 0x1234567ABABEFEF8***

***Ans 2.18.2)*** LSL X12, X11, #4 means X11 is shifted to left by 4 digits and stored in X12, i.e ,

X11 = 00100011 01000101 01100111 10000001 00100011 01000101 01100111 10000000

which can also be written as,

***X11 = 0x2345678123456780***

***Ans 2.18.3*** LSR X12, X10, #3 means X10 is shifted to right by 3 digits and stored in X12, i.e

X12 = 00000000 00000000 00000000 00000000 00010101 01010101 01010101 01010101

ANDI X12, X12, 0xFEF means (X12) **AND** (0xFEF) and store in X12, i.e,

X12 = 00000000 00000000 00000000 00000000 00010101 01010101 01010101 01010101

**AND**

**0**xFEF = 00000000 00000000 00000000 00000000 00000000 00000000 00001111 11101111

Gives,

X12 = 00000000 00000000 00000000 00000000 00000000 00000000 00000101 01000101

which can also be written as,

***X12 = 0x545***

**2.19** [10] <COD §2.6> Find the shortest sequence of LEGv8 instructions that extracts bits 16 down to 11 from register X10 and uses the value of this field to replace bits 31 down to 26 in register X11 without changing the other bits of registers X10 or X11. (Be sure to test your code using X10 = 0 and X11 = 0xffffffffffffffff. Doing so may reveal a common oversight.)

SRL X22, X10, 11

SLL X22, X22, 26

SRL X11, X11, 6

ADD X22, X22, X10

**2.20** [5] <COD §2.6> Provide a minimal set of LEGv8 instructions that may be used to implement the following pseudoinstruction: 

NOT X10, X11 // bit-wise invert

**Minimal set of LEGv8 instruction to implement the following pseudoinstruction:**

nor $X10, $X11, $zero

**2.21** [5] <COD §2.6> For the following C statement, write a minimal sequence of LEGv8 assembly instructions that performs the identical operation. Assume X11 = A, and X13 is the base address of C. 

A = C[0] << 4;

First Step is get C[0] , then Multiply by 24 and then assign into A

**LDUR   X9,[X13,#0]  
LSL X11 X9 4  
  
Explanation : LDUR will load C[0] into X9 then  
LSL X11 X9 4 will do x9 << 4 into X11 which is A**

1. Look at the following 6502 code. Compare it to the flow chart in the text book (*Figure 3.3.1*). For the five blocks (not counting Start and Done), match up the code snippets that constitute each block. (*15 pts., 3pts ea.*)

            Which code snippet matches up with “Test Multiplier0”?

  BEQ  Done if Y is equal to Zero then skipped to break branch

            Which code snippet matches up with “Add multiplicand … register”?

  ADC $700

Which code snippet matches up with “Shift Multiplicand register left 1 bit”?

  ROL  A    ; shift mltplcnd left

            Which code snippet matches up with “Shift Multiplier register right 1 bit”?

  ROR  $704      ;get bit 0 of mltplr into Carry

            Which code snippet matches up with “64th repetition? (for the 6502, eighth repetition?)”?

 jmp loop

Graphical user interface, application, table

Description automatically generated

 ; 8-bit x 8-bit unsigned multiply, 16-bit product

; mltplr in #$0704

; mltplcnd in A

; temp A in $0702

; extended A in $0703

; lo-byte product in #$0700

; hi-byte product in #$0701

LDA #$00

STA $700        ;clear lo-byte of prod

STA $701        ;clear hi-byte of prod

STA $703        ;clear extended A

LDA #255       ;multplcnd

LDX #255       ;mltplr

STX $704        ;store mltplr

LDY #9                       ;number of bits + 1 in mltplr and mltplcnd

loop:

DEY

BEQ  Done      ; if Y = 0, then done

                        ; else continue

CLC

ROR  $704      ;get bit 0 of mltplr into Carry

BCC  loop3     ;if 0, then shift only

                        ;else, add and shift

CLC                 ;clear carry for adc

STA $702        ;save lo-byte mltplcnd

ADC $700       ;add mltplcnd to lo-byte of prod

STA $700

LDA $703

ADC $701

STA $701

LDA $702

loop3:

ROL  A                        ; shift mltplcnd left

ROL  $703

loop4:

jmp loop

Done:

BRK