Assignment-4 Assembly Programming Problems

Full Credit: 20 pts | Due: Sunday 3/13

In this assignment you are asked to translate C code to ARM64/ARMv8 assembly code, or trace assembly code. C code variables should be mapped to allocated stack frame space or ARMv8 registers in assembly. You must add comment to each line of assembly code. You can test your code with Dr. Shuqun Zhang's AMR64 Online Simulator.

Q1. Translate the following C-code snippets into ARM64/ARMv8 assembly-code snippets. You must add comment to each line of assembly code to explain what it does.

1). Arithmetic operation

MOV R0,#2 ; int a=2 . (R0=a=2).

MOV R1,#-3 ; int b=-3 (R1=b=-3).

MOV R2,#10 ; int c=10 (R2=c=10).

MOV R3,#13 ; int d=13 (R3=d=13).

MUL R4,R0,R1 ; R4=R0*R1 (u=a*b)

; the next codes for division. 6 lines.

MOV R5,R3 ;copy R3=d to R5 (d value is needed further,so to calculate division move the value to R5.)

MOV R6,#0 ; R6=0, R6=v, used to hold the result orb quotient of division.

L1: SUB R5,R5,R0 ; R5=R5-R0 (d=d-a).

ADD R6,R6,#1 ; R6=R6+1 (After each subtraction increment R6 by 1 to get the result in R6.).

CMP R5,R0 ; compare R5 and R0 (d and a).

BGE L1 ; if R5>=R0 (d>=a) then goto L1 to repeated subtraction.

; End of division and result of division in R6=v.

EOR R7,R2,R3 ; R7=R2 XOR R3 (w=c^d)

AND R8,R2,R3 ; R8=R2 &R3 (x=c&d).

MOV R9,R1,LSL #3 ; R9=R1<<3 (R1 logical left shift by 3 bit and store result in R9=y)(y=b<<3).

MOV R10,R2,LSR #2 ; z=c>>2 (R2 logical shift right by 2 bit and store result in R10=z).

2). Branch and if statement.

MOV R0,#20 ; int a=20, (R0=a=20).

CMP R0,#3 ; compare R0 and 3 (condition of if statement.)

BNE else ; if a!=3 then goto else.

SUB R2,R0,#1 ; if a==3 then R2=R0-1 (b=a-1).

B exit ; branch to exit (complete if condition true statement.)

else: ADD R2,R0,#1 ; if a!=3 then else part, R2=R0+1 (b=a+1).

exit: ; end of the program.

3). Simple while loop : sum of all odd numbers that are less than 50.

MOV R0,#0 ; int sum=0(sum=R0).

MOV R1,#50 ; int n=50 (n=R1).

MOV R2,#1 ; int i=1 (i=R2).

while: CMP R2,R1 ; compare R2 and R1 to check the while condition.

BGE endwhile ; if R2>=R1,then end of while loop (while loop condition become false.)

ADD R0,R0,R2 ; if R2<R1, then R0=R0+R2 (sum+=i).

ADD R2,R2,#2 ; Then R2=R2+2 (i+=2).

B while ; branch to label while. (inside while loop.)

endwhile: ; code out of while loop.

Q2. Read the following ARM64/ARMv8 assembly-code snippets to see what it does. To understand the functionality of the code snippet, please add comment to each line.

1) Branch

```
        sub     sp, sp, #16.          updating sp = sp - 16
        mov     w0, 34                 put 34 into w0
        str     w0, [sp, 8]            store value in memory
        mov     w0, 17                w0 = value stored at memory address at sp
        str     w0, [sp, 4]           store the value from w0 to memory address at sp+4
        mov     w0, 48
        str     w0, [sp]              store the value from w0 to stack pointer
        ldr     w0, [sp, 8]
        str     w0, [sp, 12]          store the value from w0 to memory address at sp+12
        ldr     w1, [sp, 12]          load the value from memory address stored at sp+12 into
register w1

        ldr     w0, [sp, 4]           load the value from memory address stored at sp+4 into
register w0

        cmp     w1, w0                compare w0 with w1
        ble     .L2                   It means (!= , >= , <= , < , >)
        ldr     w0, [sp, 4]           load the value from memory address stored at sp+4 into
register w0

        str     w0, [sp, 12]          store the value from w0 to memory address at sp+12
.L2:
        ldr     w1, [sp, 12]          load the value from memory address stored at sp+12 into
register w1

        ldr     w0, [sp]              load the value from memory address stored at sp into
register w0

        cmp     w1, w0                compare w0 with w1
        ble     .L3                   It means (!= , >= , <= , < , >)
        ldr     w0, [sp]              load the value from memory address stored at sp into
register w0
        str     w0, [sp, 12]          store the value from w0 to memory address at sp+12
.L3:
        mov     w0, 0                 0 = value stored at memory address at w0
       add     sp, sp, 16.           updating sp = sp + 32
```

```
sub sp, sp, #32                  // updating sp = sp - 32
str wzr, [sp, 28]                // store the value(Word) from Zero Register to memory
address at sp+28
mov x0, #0x1718                  // put 0x1718 into x0
movk x0, 0x1516, lsl 16 // move 16 bit immediate to register x0 after left shift of
0x1516 by 16
movk x0, 0x1314, lsl 32 // move 16 bit immediate to register x0 after left shift of
0x1314 by 32
movk x0, 0x1112, lsl 48  // move 16 bit immediate to register x0 after left shift of
0x1112 by 48
str x0, [sp]                     // store the value(Word) from x0 to memory address at sp
mov x0, sp                       // x0 = value stored at memory address at sp
str x0, [sp, 16]                 // store the value(Word) from x0 to memory address at
sp+16
str wzr, [sp, 28]                // store the value(Word) from Zero Register to memory
address at sp+28

b .L2                            // It means go directly to branch with label L2

.L3:                             // Branch L3
ldr x0, [sp, 16]                 // load the value(Word) from memory address stored at
sp+16 into register x0
ldrb w0, [x0]                    // load the value(Byte) from memory address stored in x0
into w0
strb w0, [sp, 15]                // store the value(Byte) from w0 to memory at address
sp+15
ldrb w0, [sp, 15]                // load the value(Byte) from memory address stored at
sp+15 into w0
lsl w0, w0, #2                   // w0 = left shift w0 2 times
strb w0, [sp, 15]                // store the value(Byte) from w0 to memory at address
sp+15
ldr x0, [sp, 16]                 // load the value(Word) from memory address stored at
sp+16 into register x0
```

```
ldrb w1, [sp, 15]           // load the value(Byte) from memory address stored at
sp+15 into w1
strb w1, [x0]               // store the value(Byte) from w1 to memory at address x0
ldr x0, [sp, 16]            // load the value(Word) from memory address stored at
sp+16 into register x0
add x0, x0, 1               // x0 = x0 +1
str x0, [sp, 16]            // store the value(Word) from x0 to memory address at
sp+16
ldr w0, [sp, 28]            // load the value(Word) from memory address stored at
sp+28 into w0
add w0, w0, 1               // w0 = w0 + 1
str w0, [sp, 28]            // store the value(Word) from w0 to memory address at
sp+28

.L2:                        // Branch L2
ldr w0, [sp, 28]            // load the value(Word) from memory address stored at
sp+28 into w0
cmp w0, 7                   // compare w0 with 7 and Control status
register(CSR) stores (w0==7) , Or (w0>7) , or (w0<7)
ble .L3                     // It means (!= , >= , <= , < , >)

mov w0, 0                   // w0 = 0
add sp, sp, 32              // updating sp = sp + 32
```

<u>Submission Instructions</u>: Place all your answers into a single file, name it as **LastFirstInitial_a4.pdf**, which must include all required comments and explanations, and then upload them through your blackboard account by the due day.