ARM extra workshop - Solutions!

Anas Shrinah

December 2, 2022

Solutions!

The goal of this worksheet

This workshop aims to help you write simple ARM assembly programs.

1 Typical programing blocks in Assembly

1.1 If statement

Write an ARM assembly code to compare the values of two variables x and y; if x is equal to y, increase x by the value of y. Use r0 for x and r1 for y. Do not use any other registers.

```
if(x == y){x += y;}
```

Soultion.

```
cmp     r0, r1
addeq   r0, r0, r1
```

1.2 If...else statement

Write an ARM assembly code to compare the values of two variables x and y; if x is not equal to y, assing 3y to x or else assign 15x to y. Use r0 for x and r1 for y. Do not use any other registers.

```
if( x != y )
    {
      x = 3 * y;
    }
   else
   {
     y = 15 * x;
   }
```

Soultion.

```
cmp     r0, r1
addne   r0, r1, r1, lsl #1
rsbeq   r1, r0, r0, lsl #4
```

1.3 For loop

Write an ARM assembly code to impliment the below C++ statments. Assume r0 is c, r1 is a, r2 is b and r3 is i.

```
int c =0;
for( int i = a; i < b; i++ ){
      c++;
   }
```

Soultion.

```
mov r0, #0 @ c = 0
mov r3, r1 @ i = a
cmp r3, r2 @ compare i and b
bgt _end @ if i grater than b
beq _end @ if i less than b
add r3,#1 @ execution reaches this instruction if i is less than b (i++)
add r0,#1 @ execution reaches this instruction if i is less than b (c++)

_end: b _end
```

1.4 Do while loop

Write an ARM assembly code to impliment the below C++ statments. Assume r0 is a, r1 is b.

```
  do {
      a = a + 1;
   }while( a < b );
```

Soultion.

```
_loop:
add r0, r0, #1
cmp r0, r1
blt     _loop
```

2 Representation of characters

Write an ARM assembly program that loads registers r0,r1, and r2 with the ASCII codes of the letters in the word "fun". Then, convert these codes to the ASCII codes of the upper case English letters of the same word by:

- Adding a constant.
- subtracting a constant.

Soultion.

```
mov r0,0x66 @ ASCII code of f in hexadecimal
mov r1,0x75 @ ASCII code of u in hexadecimal
mov r2,0x5E @ ASCII code of n in hexadecimal

add r0,0xFFFFFFE0 @ F is 0x20 before f. Hence we either have to subtract 0x20 from the ASCII code
@ 0010 0000 0x2
@ 1101 1111 1'st complement
@ 1110 0000 2's complement

add r1,0xFFFFFFE0 @ same as the case of f.
add r2,0xFFFFFFE0

sub r0,r0,0x20
sub r1,r1,0x20
sub r2,r2,0x20
```

3 Calculate the inverse of a matrix

Write an ARM assembly program that finds the inverse of 2 × 2 matrix. Load the registers r0,r1,r2, and r3 with the values of a, b, c and d. then calcuate the inverse of the martix and store its elements in r7, r8, r9 and r10.

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \rightarrow A^{-1} = \frac{1}{Det(A)} \times \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

Assume that we will be working with small enough values that there will be no integer overflow.
Hints:

1. Note the division might result in numbers with decimal points.
2. Reuse your code for calculating the determinant of 2 × 2 matrices from the lab..

Soultion.

```
mov r0, #1  @load the value 1 as a load the value 1 as a
lsl r0,r0,#8 @ change the content of a to the format 24.8. i.e 1.0
mov r1, #2 @load the value 2 as b
lsl r1,r1,#8 @change the content of b to the format 24.8. i.e 2.0
mov r2, #3 @load the value 3 as c
lsl r2,r2,#8 @change the content of c to the format 24.8. i.e 3.0
mov r3, #4 @load the value 4 as d
lsl r3,r3,#8 @change the content of d to the format 24.8. i.e 4.0

bl _Det @ call the determinate function, it will save the determinate in r4.

mov r5,#1 @ load constant one so we can calculate 1/det
lsl r5, r5, #16 @ prepare the constant 1 for the division by converting it to the format 16.16

sdiv r6,r5,r4 @ singed division, results is in format 16.8. So no need to fix the format.


mul r8,r0,r6 @ multiply a (r0) by 1/det (r6)
mov r0,r8,lsr #8 @ move the result of a * 1/det to r0 and fix the format.
mul r8,r1,r6 @ multiply b (r1) by 1/det (r6)
mov r1,r8,lsr #8 @ move the result of b * 1/det to r1 and fix the format.
mul r8,r2,r6 @ multiply c (r2) by 1/det (r6)
mov r2,r8,lsr #8 @ move the result of c * 1/det to r2 and fix the format.
mul r8,r3,r6 @ multiply d (r3) by 1/det (r6)
mov r3,r8,lsr #8 @ move the result of d * 1/det to r3 and fix the format.



ldr r8,=0xFFFFFF00 @load -1 in the format 24.8 i.e. -1.0
mul r9,r1,r8 @ calcl -b: b * -1.0
mov r1,r9, lsr #8 @ fix the result format and load it to r1.
mul r9,r2,r8 @ calcl -c: c * -1.0
mov r2,r9, lsr #8 @ fix the result format and load it to r1.

@swap a (r0) and d (r3), use r8 as temp register.
mov r8,r3
mov r3,r0
```

```
mov r0,r8



_end b _end:


_Det:
mul r4,r0,r3
lsr r4,r4,#8
mul r5,r1,r2
lsr r5,r5,#8
sub r4,r4,r5
mov pc, lr
```