Advanced-CO-and-CA

Lab Assignment-3

Submitted by CS18M517

Pavan Kumar G

Problem Statement:

Finding the 32-bit number that has the largest Weight (Submission Deadline:13/10/19)

Weight of a 32-bit number is defined as **the total number of bits set**.

Given a series of 32-bit numbers (in hexadecimal form), write and assembly program to determine which element ,in the series, has the largest weight and store the number in **NUM** and its weight in **WIEGHT**.

Assume that the memory locations starting at address ***data_start*** contains the give set of integers.

For example, the *.data* section will look like

```
.data
data_start: 0x205A15E3 ;(0010 0000 0101 1010 0001 0101 1101 0011 – 13)
            0x256C8700 ;(0010 0101 0110 1100 1000 0111 0000 0000 – 11)
data_end:   0x295468F2 ;(0010 1001 0101 0100 0110 1000 1111 0010 – 14)

Output: NUM     295468F2
        WEIGHT  14
```

Add necessary comments to your program for easy readability.

Solution: I solved the issue with 2 methods:

Solution1: Naïve solution

1. Initializes wt & Max element to 0
2. Finding number of I's while iterating each element

```
while(n != 0) {
  n &= (n-1);
  count++;
```

3. Compare the wt with previous max value update the wt & Max element if it is higher.

Solution 2:

Efficient method to find the ***Hamming weight*** of a number and uses 17 arithmetic operations (works without Mul instruction)

This method is better for processors with slow MUL operations

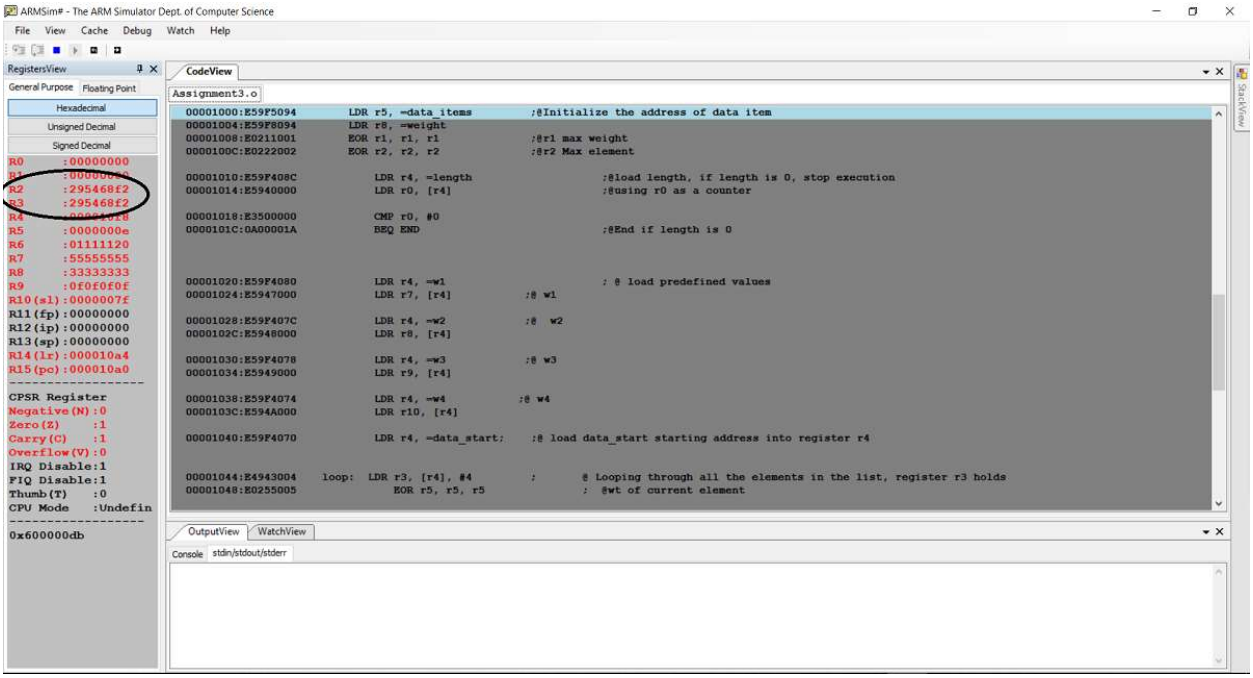
Reference: https://en.wikipedia.org/wiki/Hamming_weight

Pseudo Code:

```
int hammingWt_32bit(uint32_t x){
        x -= (x >> 1) & m1;             //put count of each 2 bits into those 2 bits
        x = (x & m2) + ((x >> 2) & m2); //put count of each 4 bits into those 4 bits
        x = (x + (x >> 4)) & m4;        //put count of each 8 bits into those 8 bits
        x += x >>  8;  //put count of each 16 bits into their lowest 8 bits
        x += x >> 16;  //put count of each 32 bits into their lowest 8 bits
        return x & 0x7f;
    }
```

1. Initializes wt & Max element to 0
2. Finding number of I's while iterating each element using ***Hamming weight***
3. Compare the wt with previous max value update the wt & Max element if it is higher.

Result:


ARMSim# - The ARM Simulator Dept. of Computer Science
File View Cache Debug Watch Help
RegistersView
General Purpose Floating Point
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 :00000000
R2 :295468f2
R3 :295468f2
R5 :0000000e
R6 :01111120
R7 :55555555
R8 :33333333
R9 :0f0f0f0f
R10(sl):0000007f
R11(fp):00000000
R12(ip):00000000
R13(sp):00000000
R14(lr):000010a4
R15(pc):000010a0
CPSR Register
Negative(N):0
Zero(Z) :1
Carry(C) :1
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T) :0
CPU Mode :Undefin
0x600000db
CodeView
Assignment3.o
00001000:E59F5094 LDR r5, =data_items ;@Initialize the address of data item
00001004:E59F8094 LDR r8, =weight
00001008:E0211001 EOR r1, r1, r1 ;@r1 max weight
0000100C:E0222002 EOR r2, r2, r2 ;@r2 Max element
00001010:E59F408C LDR r4, =length ;@load length, if length is 0, stop execution
00001014:E5940000 LDR r0, [r4] ;@using r0 as a counter
00001018:E3500000 CMP r0, #0
0000101C:0A00001A BEQ END ;@End if length is 0
00001020:E59F4080 LDR r4, =w1 ; @ load predefined values
00001024:E5947000 LDR r7, [r4] ;@ w1
00001028:E59F407C LDR r4, =w2 ;@ w2
0000102C:E5948000 LDR r8, [r4]
00001030:E59F4078 LDR r4, =w3 ;@ w3
00001034:E5949000 LDR r9, [r4]
00001038:E59F4074 LDR r4, =w4 ;@ w4
0000103C:E594A000 LDR r10, [r4]
00001040:E59F4070 LDR r4, =data_start; ;@ load data_start starting address into register r4
00001044:E4943004 loop: LDR r3, [r4], #4 ; @ Looping through all the elements in the list, register r3 holds
00001048:E0255005 EOR r5, r5, r5 ; @wt of current element
OutputView
WatchView
Console
stdin/stdout/stderr