11/21/16 Assembly

Class Notes

Single instruction multiple data

SIMD

* Def.) Simple Instruction Multiple Data

g++ flag: -O3

* -O3 optimizes the program for speed

Function for divide

* Input: r0=a, r1 = b
* Output: r1 =b
* Output r0 = a%b, r1 = a/b
* Result: r0 = 17, r1= 0
* Out: r0 = 17, r1 = 8,
* Bl div\_max;

Compiling the program

* G++ can include the other files
  + G++ main .s myfunctions –o main
    - Other Libraries:
      * -lWiringPi

Ex)

* Sequential thinking

Address calculation

* Data:
  + Int a[4] = {11,12,13,14};
  + Int b[4] = {15,16,17,18};
  + Int sum[4]={0,0,0,0};
  + Int product[4]={0,0,0,0};
* Used to locate items in memory
  + Uses loop to help iterate
    - Ex) For (int i = 0 ; i < 3; i +=2){

R4 = a[i+1];

R5 = b[i+1];

Sum[i] = r4 +r5

Product[i] r4\*r5

}

* + - Loads: 8
    - Compare: 5
    - Adds: 24
    - Mult: 20
    - Store: 8
    - Mov: 8

Registers

* R registers: r0 – r15
  + 32 bit ints
* S registers: s0 – s31
  + 32 bit float
* D registers: d0 – d15
  + 64 bit float (double)
* Q registers: Q0 – Q15
  + 128 bits
  + For ints & floats
  + Visualization:
    - [ 32 bits ] [32 bits ] [ 32 bits ] [ 32 bits ]

0 – 31 32- 63 64 – 95 96 - 127

* + - Bits don’t carry over
* Loading multiple data into Q reg.
  + Goal:
    - Get a and b into Q registries
      * Q0: {11,12,13,14}
      * Q1: {15,16,17,18}
  + Loading:
    - Ex) Ldr r0, =a

//Vecter load 1 item, 32 bits at a time

//This line loads all the items in array into one register

VLD1.32 {Q0}, [R0] //Instruction can be rewritten \*\*

//Loads array b into Q1

Ldr r0, =b

VLD1.32 {Q1}, [R0]

* + - \*\* Rewrite:

//Items in memory remain, even after altered

//Loads the first 128 bits into Q0 then continues through memory to load Q1 //with array b

VLD1.32 {Q0, Q1} [R0]

* + This method requires less loads, comparisons, etc. since assembly performs multiple actions in one line of code.
    - Ie. SIMD

Vector functions

* Vadd.u32
  + Adds the values within vectors by increments of unsigned 32 bit values
* Vmul.u32
  + Multiplies the values within vectors by increments of unsigned 32 bit values
* Vst1.32
  + Store a vector of data one value at a time by increments of 32 bits

In order to use SIMD keywords

* Include in code: .fpu neon