**CS 2160 Final**

**Summer 2022**

1. (10 pts) Write a short descrip3on of the x86 architecture that would be suitable to provide a programmer who is unfamiliar with x86\_65. (No need to describe the FP or SIMD features)

X86\_64 is an CISC architecture that defines a word as 16 bits, with 16 general purpose registers (64 bits) with different names than the AArch architectures and some of the instructions having implied operands (see below) and depending whether or not you are using linux or a Microsoft based system, has different ways to pass values in for operations.

1. (5 pts) What is an “implied operand” and how does it relate to the x86\_64 architecture?

An implied operand is exclusive to x86\_64 where for some operations (like div) one of the operands HAS to be in a specific register as that “implied” one is the one the architecture uses to operate.

1. (5 pts) Describe either the GNU or MicrosoQ calling convention for the C language and the x86\_64 architecture. (Specify which convention you’re describing)

GNU: For gnu, when C passes a called assembly program to x86\_64, it can pass up to 6 values into specific registers that are rdi, rsi, rdx, rcx, r8 and r9 and the rest get placed onto the stack

Microsoft: Microsoft when passing can only send 4 into registerd ecx, edx, r8d and r9d and then puts things onto the stack

1. (5 pts) Briefly describe the func3on of the following:
   1. Assembler = compiles assembly programs and writes them to an object file
   2. Linker = connects different object files and makes them executable
   3. Loader = part of the OS that loads and execute said executable files
2. (5 pts) What is the size of a *double* w*ord* in x86\_64?

In x86\_64, the size of a double word is 32 bits.

1. (5 pts) What does the RIP do in the x86\_64 architecture? What does it point to?

The regitster, instruction pointer (RIP) in x86\_64, like the IP in AArch systems, points to the address of the next instruction to be performed.

1. (5 pts) To reference memory, what four inputs can be used and how do they relate to the memory address generated?

For memory there are 2 direct ways to access it and 2 indirect ways to access it. The Direct ways to access it is to use the actual memory address listed in hex, you can also store that address in a register to access and then dereference the register. Indirectly you can Save a variable to the register and dereference again to access the memory address of the variable. Or you can directly put the address of the variable into the register and use it the same as before.

1. (5 pts) What does SIMD mean in computer architecture.

SIMD means Single Instruction, multiple Data meaning it can perform one instruction on multiple sets of data inputs in parallel.

1. (10 pts) As a computer programmer, would you prefer to write code for the AArch32 or x86\_64 architecture? Why?

I would prefer to use AArch32 because it is a little less complex, having no implied operands and being a RISC system and not a CISC system. Therefore it is easier to begin to use and less to remember with similar functionalities (on a lesser scale since it is 32bit).

1. Given the following C program, write assembly rou3nes for:
   1. (15 pts) int isOdd(int a)
   2. (15 pts) int someCalcs(int a, int b, int c)
   3. (15 pts) int arrRef1d(int \*arr, int element)

#include <stdio.h>

/\*

Function isOdd returns 1 if the argument a is an

odd integer, 0 otherwise

\*/

extern void isOdd (int a) asm ("intMean\_");

/\*

Function someCalcs returns (a + b) / c using integer division.

If c == 0, returns –1

\*/

extern int someCalcs (int a, int b, int c)asm

(“someCalcs\_”)

/\*

Function arrRef1d returns arr[element]

\*/

extern int arrRef1d) (int\* arr, int element) asm

(“arrReff1d\_”)

void main () {

int arr[10] = {1, 2, 3, 4, 5}; int x = 10, y = 20, z = 30; a = 40;

printf(“Results of isOdd(5): %d \n”, isOdd(5)); printf(“Results of isOdd(10): %d \n”, isOdd(10));

printf(“a = %d, b = %d, c = %d, result = %d\n”, x, y, z, someCalcs(x, y, z));

printf(“a = %d, b = %d, c = %d, result = %d\n”,

1, 2, 0, someCalcs(1, 2, 0));

printf(“arr[2] = %d\n”, arrRef1d(arr, 2)); printf(“arr[7] = %d\n”, arrRef1d(arr, 7));

printf(“Thanks for a great summer semester!\n”);

}