**Student Name:** Hatice Nur Ağba

**Student Number:** 110510039

# COMP 305 – HW2

**Readme:**

1. Make sure that your matmul.c and matmul.h works properly
2. Type make ins.so TARGET=intel64
3. Type ../../../pin -t obj-intel64/ins.so -- /your_executable_for_matmul
4. In order to view the result, type cat ins.out

**Task1:** Count the total number of instructions in the application. Also answer if this number is for macroinstructions or microinstructions.

**Answer:**

When I run the application with following steps;

H – help

M – Multiply

Q – Quit

Y – Confirm Quit

The total number of instructions are: 907684

In order to achieve this result, ins.cpp calls docount method for every instruction in the application to count the total instruction in runtime.

**Task2:**

In the ISAs, there are three classes of machine instructions:

- Memory Instructions (ie. Loads and Stores)
- Branch Instructions (ie. Jumps, Branches etc.)
- Arithmetic and Logic Instructions (ie. Add, Sub, Mul, Div, Shift, And, Or etc.)

Count the number of instructions in each class in the matrix multiplication application.

**Answer:**

When I run the application with following steps;

H – help

M – Multiply

Q – Quit

Y – Confirm Quit

Memory Instructions: 364742

Branch Instructions: 86804

Arithmetic/Logic Operations: 456138

In order to achieve this result, ins.cpp sends 0 for memory instructions, 1 for branch operations and 2 for arithmetic operations to the docount method as argument in order to count every type of instruction separately.

Since these are directly related to hardware, they are micro instructions.

**Task3:** Here is the list of registers and their purposes in X86:

The purposes of each register are as follows:

RAX: Accumulator

RBX: Base index (for use with arrays)

RCX: Counter (for use with loops and strings)

RDX: Extend the precision of the accumulator

RSI: Source index for string operations.

RDI: Destination index for string operations.

RSP: Stack pointer for top address of the stack.

RBP: Stack base pointer for holding the address of the current stack frame.

R8-R15: general purposed registers

Calculate the frequency of usage of each register as a destination register in your application.

**Answer:**

When I run the application with following steps;

H – help

M – Multiply

Q – Quit

Y – Confirm Quit

The frequency of destination registers that are used as follows;

si: 6
r10b: 2
esi: 144
r10d: 39
ah: 13
r11d: 26
al: 94
st0: 2
r12b: 2
ymm0: 18
r12d: 47
ymm1: 16
bpl: 4
ymm2: 9
bl: 6
ymm3: 1
sil: 27
r13b: 4
ymm4: 1
ch: 12
r13d: 61
cl: 22
r14b: 4
ymm9: 8
dh: 18
ax: 20
ebp: 48
r14d: 30
dl: 80
edi: 94
eax: 809
r8b: 13
r15b: 16
r11w: 4
r8d: 67
dil: 2
r15d: 53
ebx: 84
r9b: 6
cx: 2
r9d: 77
ecx: 165
rsi: 478
dx: 9
r10: 129
r11: 55
r12: 357
edx: 362
r13: 310
r14: 419
fs: 54
r8: 112
r15: 309
r9: 106
rsp: 1891
rflags: 46
rbp: 1367
rdi: 700
rax: 1782
rbx: 1059
rcx: 470
rdx: 896
xmm0: 160
xmm1: 88
xmm2: 116
xmm3: 26
xmm4: 12
xmm5: 16
xmm6: 3
xmm8: 6
xmm9: 1
rip: 5234

In order to achieve this result, ins.cpp uses a map to store register names and their usage counts. For every instruction, it gets its write register and increase its value by one in the map.