**Introduction**: An important part of this class is to research and present some interesting area of Computer Organization and Architecture. The class will be divided into groups of four students. Each group must work together on this project as a team. Each group member is expected to be in class the day of their group’s presentation or a late penalty will be applied to the missing person.

A 45 – 60 minute PowerPoint presentation to the class is required. Points will be reduced for presentations that are too short. This time period **does not** include the Question and Answer session at the end of each presentation.

**Presentations:** All presentations must be in PowerPoint. Other pertinent items may be included in the discussion. (e.g. A peripheral presentation on disk drives may include a disk drive to pass around the class. Working code examples would be helpful.) All presentations must begin with team introductions stating assigned tasks per team member, follow by project introduction of material to be discussed. Work assignments and roles must be declared at the start of your presentation. All presentations must end with a summary of topics with a final slide listing References. A question and answer session will follow the presentation.

The final presentation PowerPoint slides must be submitted to the D2L Project drop box before the day of the presentation to avoid a group late penalty.

**Participant Responsibilities**: Each member of the team is required to equally participate in the research, creation and presentation of the presentation. Each member is required to present a portion of the presentation and participate equally in the question and answer period.

As a guideline since there are about 4 students in each group, each student should present for about 10 - 15 minutes and discuss 10 – 15 slides depending on how much information is on each slide. Use your best judgement here. There are no stringent rules.

Project will be graded as follows:

(100 pts) Project Presentation (45 – 60 minutes)

Team Introduction and roles of each person

Project Introduction: Project Description and Discussion Points

Project Main Presentation: Detailed Project Discussion Points

Project Summary: A review of material covered in the Presentation

Demonstration and/or Supporting Materials

Question and Answer session

(100 pts) PowerPoint Slides

Presentation slides clarity and completeness

Presenter Notes and Comments section of the PowerPoint Slides

Knowledge Depth

The grading of this project is somewhat subjective in that the quality of presentations may vary greatly. Grading will be done after all groups have presented. In general everyone will start with 180 out of the 200 points. If your presentation is better than the rest additional points will be added. If your presentation is not as good as the rest some points will be deducted.

**Group Project Topic 1**: Intel 64-bit Processor

Short history of Intel processors

CPU architecture including cores, hyper-threading and caching.

CPU memory management architecture including paging and virtual memory.

Instruction set architecture including registers, sample instructions and stack management.

Sample program written in C (with assembly explanations) compiled both with and without optimization.

**Group Project Topic 2**: ARM 64-bit Processor

Short history of ARMv8 processors

Sample of ARMv8 implementations (hardware vendors)

CPU architecture including cores, hyper-threading and caching.

CPU memory management architecture including paging and virtual memory.

Instruction set architecture including registers, sample instructions and stack management.

Sample program written in C (with assembly explanations) compiled both with and without optimization.

**Group Project Topic 3**: RISC-V Processor

Short history of RISC-V processors

Sample of RISC-V implementations (hardware vendors)

CPU architecture including cores, hyper-threading and caching.

CPU memory management architecture including paging and virtual memory.

Instruction set architecture including registers, sample instructions and stack management.

Sample program written in C (with assembly explanations) compiled both with and without optimization.

**Group Project Topic 4**: Java Virtual Machine (JVM)

Short history of Java

Java Class file structure and bytecode

Java Memory Management

Java implementations – Platforms / vendors

What other languages compile to Java Bytecode

Android Implementation

JIT compile/execution

Performance comparison between C/C++ and Java with and without JIT

Sample Java program with explanation of bytecode

**Group Project Topic 5**: I/O and Storage Subsystems

Types of I/O devices

Bus protocols: IDE, SAS, SATA

Details on Disk Drives

USB and Plug and Play

**Group Project Topic 6**: Networking

TCP/IP/Ethernet layered network communication structure (relate to OSI model)

IP Packet Assembly/Disassembly

TCP Handshaking, Windowing, Keep Alive, Flow Control

TCP vs UDP

IP address structure (IPv4 vs. IPv6)

DHCP and DNS address protocols

Common protocols including FTP, POP, IMAP, and HTTP with some details on the commands

Wireshark traces of an actual FTP or HTTP connection with an explanation of each packet

The online Chapter 12 from our textbook could be an excellent resource for this project

**Group Project Topic 7**: Compilers, Assemblers, and Linkers

What does a compiler do?

What is optimization? Describe various types of optimizations. Provide some examples.

What does an assembler do?

What does a linker do?

What is an executable?

What is a shared library / DLL?

What is dynamic linking/loading? How are symbols resolved?

Discuss differences between GCC and LLVM.

**Group Project Topic 8**: Quantum Computing

What is quantum computing?

What problems can be solved with quantum computing?

History of quantum computing.

What is a qubit? How many qubits does a quantum computer need to be useful?

What are the leading companies? What are their goals? How do they differ?

Describe in detail the components of a quantum computer.

How is a quantum computer programmed?

What are the technology issues that need to be resolved?

What are some of the responses as to how cryptography can still be safe?

**Group Project Topic 9**: Cloud Computing

What is cloud computing?

What problems can be solved with cloud computing?

What are the leading companies?

For each company what cloud computing models to they offer?

What is the purpose of each model and how do they differ?

Describe in detail the how a cloud computing environment is structured?

How are the data centers structured and where are they located?

How is reliability and performance guaranteed? How do new virtual machines come into existence?

**Notes on Group Project Topics 1 – 3:**

1. On Windows 10 you can use the *Windows Subsystem for Linux* to compile / assemble programs. The following URL may help installing WSL 2:

<https://docs.microsoft.com/en-us/windows/wsl/install-win10>

You can install any Linux distribution you like. I used Ubunto.

1. Use sudo apt update to update the package database.
2. Use sudo apt upgrade to make sure all of your packages are current.
3. If the project is Intel 64 bit:
   1. Use sudo apt install gcc to install the GNU C x86 and x86-x64 compiler.
   2. Use sudo apt install gcc-multilib to install the GNU C cross- compilation feature.
   3. Compile with gcc -m64 for 64-bit compiles.
4. If the project is ARM 64 bit:
   1. Use sudo apt install gcc-aarch64-linux-gnu to install the GNU C compiler ARM 64.
   2. Compile with aarch64-linux-gnu-gcc.
5. If the project is RISC-V:
   1. Use sudo apt install gcc-riscv64-linux-gnu to install the GNU C compiler RISC-V.
   2. Compile with riscv64-linux-gnu-gcc.
6. To compile an unoptimized version of your C program use:

<gcc-compiler-name> -Wa,-adhln -g -masm=intel sample.c > sample-g.asm

This will write the unoptimized inter-mixed source and assembly code to sample-g.asm. The option -masm=intel should only be used with the Intel project.

1. To compile an optimized version use:

<gcc-compiler-name> -Wa,-adhln -O -masm=intel sample.c > sample-o.asm

This will write the optimized inter-mixed source and assembly code to sample-o.asm. The option -masm=intel should only be used with the Intel project.

**Sample C program you could use for Group Project Topics 1 – 3. This sample program may too long to discuss in detail. Feel free to simplify or only discuss portions of the generated assembly language code.**

/\* ICS 232 - Sample C program to show assembly code \*/

/\* Robin Ehrlich \*/

/\* 2020/01/18 \*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

int function1(int x)

{

int y;

x += 1;

y = 10 \* x;

return (y);

}

int function2(int \*x)

{

int y;

\*x += 1;

y = 10 \* \*x;

return (y);

}

int function3(int \*num, int len)

{

int i;

int max;

max = num[0];

for (i = 1; i < len; i++) {

if (max < num[i])

max = num[i];

}

return (max);

}

int main(int argc, char \*\* argv)

{

int i;

int num[10];

int r;

for (i = 1; i < 5; i++) {

r = function1(i);

printf("i = %d, function1 = %d\n", i, r);

}

for (i = 1; i < 5; i++) {

r = function2(&i);

printf("i = %d, function2 = %d\n", i, r);

}

srand(time(NULL));

for (i = 0; i < 10; i++) {

num[i] = rand();

printf("Random %d = %d\n", i, num[i]);

}

r = function3(num, 10);

printf("function3 = %d\n", r);

return (0);

}