**Developing Soft and Parallel Programming Skills Using Project-Based Learning**

**By Assembly Chefs**

**Spring 2020**

**Team Coordinator:**

**Rahul Sunkara**

**Members:**

**Andy Lee**

**Bryanna Hardy**

**Mezemir Gebre**

**Nathan Heckman**

Table of Contents

Introduction:

Project Cover Page pg. 1

Table of Contents pg. 2

Scheduling and Planning pg. 4

Rahul Sunkara’s Reports:

Rahul’s Title Page pg. 5

Parallel programming foundations pg. 6

Parallel programming report pg. 8

ARM assembly programming pg. 9

Andy Lee’s Reports:

Andy’s Title Page pg. 11

Parallel programming foundations pg. 12

Parallel programming report pg. 14

ARM assembly programming pg. 16

Bryanna Hardy’s Reports:

Bryanna’s Title Page pg. 19

Parallel programming foundations pg. 20

Parallel programming report pg. 21

ARM assembly programming pg. 22

Nathan Heckman’s Reports:

Nathan’s Title Page pg. 24

Parallel programming foundations pg. 25

Parallel programming report pg. 27

ARM assembly programming pg. 30

Mezemir Gebre’s Reports:

Mezemir’s Title Page pg. 35

Parallel programming foundations pg. 36

Parallel programming report pg. 37

ARM assembly programming pg. 39

Appendix:

Links to Slack, GitHub, YouTube pg. 42

Screenshots pg. 43

Scheduling and Planning

|  |  |  |  |
| --- | --- | --- | --- |
| **Member’s name** | **Email** | **Task** | **Due Date** |
| Rahul Sunkara | rsunkara3@student.gsu.edu | Create Team Schedule, summarize the final report, Parallel programming basics report, ARM assembly program coding and report. | 2/18/20 |
| Andy Lee | Alee162@student.gsu.edu | Parallel programming basics report, ARM assembly program coding, help summarize the final report. | 2/18/20 |
| Bryanna Hardy | bhardy11@student.gsu.edu | Parallel programming basics report, ARM assembly program coding and report. Adding the TA into Slack. | 2/18/20 |
| Nathan Heckman | nheckman1@student.gsu.edu | Record and Edit presentation video, Parallel programming basics report, ARM assembly program coding and report. | 2/18/20 |
| Mezemir Gebre | mgebre1@student.gsu.edu | Parallel programming basics report, ARM assembly program coding and report. Creating folders for GitHub. | 2/18/20 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Member’s name** | **Duration** | **Dependency** | **Note** |
| Rahul Sunkara | Meeting: 1:30 hours  Task: 2:00 hours | Microsoft Words, Raspberry Pi, GitHub | Done All Task  100% Grade |
| Andy Lee | Meeting: 1:30 hours  Task: 1:30 hours | GitHub, Microsoft Word, Slack, Raspberry Pi | Done All Task  100% Grade |
| Bryanna Hardy | Meeting: 1:30 hours  Task: 2:15 hours | GitHub, Raspberry Pi, Microsoft Word | Done All Task  100% Grade |
| Nathan Heckman | Meeting: 1:30 hours  Task: 2:30 hours | Movie Editor, Raspberry Pi, Microsoft Word, GitHub | Done All Task  100% Grade |
| Mezemir Gebre | Meeting: 1:30 hours  Task: 2:30 hours | GitHub, Microsoft Word, Raspberry Pi | Done All Task  100% Grade |

**Summary:**

The project assignment 2 required everyone to communicate with each of the team members and collectively/ successfully achieve the final product. The crucial assignments for Project A2 were to: create a schedule table, complete the parallel programming foundations questions and the basis, invite the TA into Slack and create required folders in GitHub, to code and report the ARM assembly; lastly, to record and upload a presentation video. All the tasks were distributed equally among the members, except the parallel programming foundations, parallel programming basics, and the ARM assembly report.

As mentioned in the schedule/ planner above, Rahul Sunkara, the team coordinator, had the responsibility of creating the planning table and making the final report. Andy Lee took up the responsibility to answer the parallel programming foundations questions along with Bryanna Hardy, who also had the responsibility of adding the TA into Slack and Mezemir Gebre. Nathan Heckman had the responsibility of making a presentation video.

**Rahul Sunkara Individual Questions and Reports**

Parallel Programming Foundations

Task 3

1. **Identifying the components on the raspberry PI B+**

CPU/ RAM, Ethernet, USB, Ethernet Controller, Display, HDMI, Camera,

1. **How many cores does the Raspberry Pi’s B+ CPU have?**

Raspberry Pi’s B+ CPU has 4 cores

1. **List three main differences between X86 (CISC) and ARM Raspberry PI (RISC). Justify you answer and use your own words (do not copy and paste)**

A few of the main differences between x86 and ARM are that: 1) x86 is a CISC (complex instruction set computing) which allows the x86’s processor to have a bigger feature rich instruction set that allows multiple ways to access the memory, but it has far less registers than ARM. 2) ARM is a RISC (Reduced instruction set computing) which is a much more simplified instruction set and has more general registers that use load/store to access memory. 3) Most instruction used in ARM programming can be used for conditional execution; Intel x86’s processors use the little-endian format

1. **What is the difference between sequential and parallel computation and identify the practical significance of each?**
   1. Sequential computations processes are executed in a continuous/ ordered process one after the other, while the parallel computation processes are executed simultaneously.
   2. There is no need to change a sequential computation when using and if it is being used in different modules it does not require any communication at the module interfaces. Parallel computation on the other hand has the ability to be converted to a sequential composition so that it can enhance scalability and locality.
2. **Identify the basic form of data and task parallelism in computational problems.**

Data parallelism uses multiple processing elements to solve problems by breaking the problem into independent parts. Task is a form of parallelization of computer code across multiple processors in the parallel computing environments.

1. **Explain the differences between processes and threads.**

The difference between processes and threads, processes is the abstraction of a running program, and they do not share a memory with each other. A thread is a lightweight process that allows a single executable/process to be decomposed to smaller, independent parts. Every thread shares common memories of the processes they belong to.

1. **What is OpenMP and what is OpenMP pragmas?**

OpenMP is a library/language. OpenMP pragmas are compiler directives that enable the compiler to generate threaded code.

1. **What applications benefit from multi-core (list four)?**

Database servers; Web servers; Compilers; Multimedia applications; Scientific applications; In general, applications with thread-level parallelism

1. **Why Multicore? (why not single core, list four)**

Multicore is better because it is difficult to make single-core clock frequencies higher, and many new applications are also multithreaded, which currently is a trend in computer architecture

Parallel Programming Report

A screenshot of a computer

Description automatically generated

This image shows the threads sharing the same fork/thread because the code is not correct. Also, the threads are not in order, even if they share threadsA screenshot of a cell phone

Description automatically generated A screenshot of a computer

Description automatically generated

The images above show the corrected code and the result after the threads are printed out. The threads now have independent fork/ threads but are still out of order because once a thread is finished and printed, the order does not matter.

ARM assembly Programming

Task 4

**Part A**

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

The above images are the registers with the solution and the code of the tutorial ARM Assembly program. This tutorial gave me a better understanding of loading memory into the registers through the variables that are already in .data. The coding in ARM is different compared to x86 because in ARM we need to load the address of the variable first and then load the actual value. After loading the address and values the actual arithmetic is calculated and the result is stored, which is 7.

A screenshot of a cell phone

Description automatically generated**Part B**

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

As stated above I first had to store the values of the variables in the .data section. Then I loaded the variable address’s and then their values from the given equation, register = val2 + 9 + val3 – val1. Later, I computed the actual arithmetic of the equation, by using the registers and the immediate value. After this I loaded the register the stored the final result into it, which is 30.

**Andy Lee Individual Questions and Reports**

Parallel Programming Foundations

Task 3

**Foundation** (reading material for this section is available at iCollege, Week4, Project:

Assignment folder, Introduction\_to\_Parallel\_Computing\_and\_the\_Raspberry\_Pi\_A2.pdf)

**Identifying the components on the raspberry PI B+**

Raspberry PI B+’s components are Power, Central Process Unit, Random Access memory, USB, Display, HDMI and Camera. It allows you to use wireless internet by connecting WIFI, but Ethernet, and Ethernet Controller are also included as the components.

**How many cores does the Raspberry Pi’s B+ CPU have?**

Raspberry Pi B+’s CPU has 4 cores.

**List three main differences between X86 (CISC) and ARM Raspberry PI (RISC). Justify you answer and use your own words (do not copy and paste)**

X86 have a CISC structure for their processor which make you to process faster by having a larger and more feature-rich instruction set. This mean the X86’s CPU would have more operations, address mode. The tradeoff for this process is that ARM would have more registers/memories than the X86.

Second, Unlike X86, ARM used RISC processor. On above paragraph, we said that the CISC has more instruction set, and we could confirm that statement since RISC has 100 instructions at most. By doing this, we could have more “general” purpose registers. Due to lack of instructions, only “load” and “store” can access to the memory.

Overall, the difference in number of instructions has pros and cons. For having less instruction, the processor would execute the program faster and efficiently. But, having less instruction means there is less tool to use, which gave a huge problem to the programmers.

**What is the difference between sequential and parallel computation and identify the practical significance of each?**

Sequential computing has a consecutive order when processing. This means the processor would compute one after another. The processor never stops until the program is terminated.

The parallel compute the program at the same time using a different thread. It computes a multiple instruction at the once, but in order to do this, we need a mutli-core CPU. Since it computes concurrently, it is way faster than sequential computing.

In general, parallel are used in database, and data mining where lots of data is being computing, and sequential usually done when programs needs to be safely (and stable) compute

**Identify the basic form of data and task parallelism in computational problems.**

Data parallelism is an execution of a data assigned by the computer. In works concurrently, but it could compute the different functions across the processor.

The task parallelism executes the same function on multiple cores. It should be fast to compute since we are using a equitant functions to compute different data.

**Explain the differences between processes and threads.**

Processor’s running program is an abstraction on running program. This makes the process to prohibit to share the memory.

Threads are lightweight process which allows to perform a single process to be decomposed to smaller and makes into independent parts. It shares the common memories of the process.

**What is OpenMP and what is OpenMP pragmas?**

OpenMP is a library/language which the compiler’s directives enable the compiler to generate threaded code.

**What applications benefit from multi-core (list four)?**

1.Database servers

2.Web servers

3.Compilers

4.Multimedia applications

5.Scientific applications

In general, applications with thread-level parallelism

**Why Multicore? (why not single core, list four)**

The most advantage on multicore processor is that it is easier to make the CPU itself compares to the single core clock. Also, multicore would help to creates the more multithreads which increase the computer performance.

Difficult to make single-core clock frequencies higher

Many new applications are multithreaded

General trend in computer architecture (parallelism is preferred)

Parallel Programming Report

A screenshot of a cell phone

Description automatically generated

The code for this report is directly from the instruction pdf. This code will divide the tasks to individual cores, so that the computer could run efficiently. Since raspberry pi (B+) does have the 4 cores in CPU, the result of this code will be given 4 output. In order to understand this code properly, it need the basic knowledge on “omp.h” library, but for simplification, the computer would printout thread id , and number of threads. The id will give us a unique number (if we do properly), and the number of threads will be fixed .

A screenshot of a computer screen

Description automatically generated

AS you could see, the code have 4 output. The main reason why this happen is because the CPU have 4 cores. Individual cores will take the tasks, and printout the number of threads. As you could realize, the all threads are unique, which we know they we are using the thread efficiently. The program is written in c language, and we must use the c compiler to run this program.

ARM assembly Programming

Task 4

**Register info**

A screenshot of a cell phone

Description automatically generated

The result of the arithmetic is stored in a r1 register. The value of the result is 30, and we know that is the correct value

**Accessing the memory address using the commands**

A screenshot of a cell phone

Description automatically generated

We could identify the data using the address. I have used “x/3dw 0x7efff3b0” to identify which data is stored in that address, and the code shows that address ox7eff3b0 stored 1 2130703671 (I assume it is a garbage data)

**Code**

A screenshot of a cell phone

Description automatically generated

We are preforming the basic arithmetic using the register. The assign task is to calculates val2 + 9 +val3 -val1 and stored result into register. As you could see above, I have declared the global variable named as “valOne, valTwo, and valThree”. Each variable could store a numerical number, and in this case, we store 6 11 and 16. Going into the main function, we go to main stack and load all the necessary data (in this case, load valOne, valTwo, and valThree) using ldr function. Once load is done, we now can perform the arithmetic using add and sub function. The code above will show how to use a add and sub command. Last but not least, all we need to do is to store the result value into register r1. The current result value is stored into the r2, so I have done “add r1, r2, #0.”. I do aware that I could use the “str r1, r2” to store the value, but above code also work.

**Bryanna Hardy Individual Questions and Reports**

Parallel Programming Foundations

Task 3

1. Identify the components on the Raspberry Pi B+.
   1. The components of a Raspberry Pi are the ethernet, USB, ethernet controller, display, HDMI, camera, the CPU/RAM, and the power.
2. How many cores does the Raspberry Pi’s B+ CPU have?
   1. It has 4 cores.
3. List the three main differences between the X866 (CISC) and ARM Raspberry PI (RISC).
   1. The X86 is a complex instruction set computing (CISC) which allows the processor to have a bigger feature instruction set that allows multiple ways to access the memory. The ARM Raspberry PI is a reduced instruction set computing (RISC) which is a more simplified instruction set and have more general registers than the X86 which load/store to access the memory. The instructions used in ARM can be used for conditional execution. On the other hand, X86’s processors use the Little-Endian format.
4. What is the difference between sequential and parallel computation? Identify the practical significance of each.
   1. Sequential computations processes are executed in a consecutive and ordered process. Parallel computation processes are executed concurrently.
   2. Sequential can be used unchanged and if different modules use the same data distribution, no communication is really required at the module interfaces. Parallel can be converted to a sequential composition and mainly it can enhance scalability and locality.
5. Identify the basic form of data and task parallelism in computational problems.
   1. Data parallelism uses multiple processing elements to solve problems by breaking the problem into parts. Task is a form of parallelization of computer code across multiple processors.
6. Explain the differences between processes and threads.
   1. Processes is the abstraction of a running program and it does not share memory. On the other hand, a thread is a lightweight process that allows a single executable/process to be decomposed to smaller parts. Each thread shares common memories of the processes they belong to.
7. What is OpenMP and what is OpenMP pragmas?
   1. OpenMP is a library/language and OpenMP pragmas are compiler directives that enable the compiler to generate threaded code.
8. What applications benefit from multi-core (list four)?
   1. Database servers; Web servers; Compilers; Multimedia applications; Scientific applications; In general, applications with thread-level parallelism
9. Why Multicore? (why not single core, list four)
   1. Multicore is better because it is difficult to make single-core clock frequencies higher, and many new applications are also multithreaded, which currently is a trend in computer architecture

Parallel Programming Report

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generatedThese two images are before the code was corrected. The reason why each thread shares the same fork/thread is because of the code not being correct. On the other hand, as you can see in images they are not in order even if they share the same fork/thread. It is because it does not matter the order of when a thread is finished and printed. When the code is corrected, the images will be inserted down below, the threads will still not be in order, but it will have its own independent fork/thread. A screenshot of a computer

Description automatically generated

ARM assembly Programming

Task 4

**Part A**

A screenshot of a computer screen

Description automatically generatedA screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated

These images above are from the tutorial of the ARM Assembly. It taught me how to load the memory into the register(s) from using the variables that are already from the .data. This is a little bit different than coding in Intel x86. The tutorial instructed me to basically add two variables and load/store it into another variable. As you can see, in the first image, I loaded each variable into a register and once I did that, I added the two registers that hold the variables a and b, which gives you the sum of 7. Then, I store the sum into r1 and as you can see in the third image, you can see the sum stored into that register.

**Part B**

A screenshot of a cell phone

Description automatically generated A screenshot of a cell phone

Description automatically generated

A screenshot of a computer

Description automatically generated

These images are from Try It On Your Own in ARM Assembly. I used the tutorial as a template to help me create this program. As you can see, I had to code an equation, register = val2 +9 + val3 - val1. To do so, I stored each variable with a value in the .data. Once I completed that, I loaded and stored each variable into a register. After that, I added the immediate value to register 2, and then I added r1 to r2. Then, I loaded the variable register into r3, and stored the r3 into r2. In the last image, you can see that r2, has the correct answer, 30.

**Nathan Heckman Individual Questions and Reports**

Parallel Programming Foundations

Task 3

1. **Identify the components on the Raspberry Pi B+.**

Components on the Raspberry Pi B+ include the CPU/RAM, Ethernet Controller, Ethernet, USB ports, power, HDMI, camera, and display.  
**2. How many cores does the Raspberry Pi’s B+ CPU have?**

Raspberry Pi B+ CPU has 4 cores.

1. **List three main differences between X86 (CISC) and ARM Raspberry PI (RISC).**
2. X86 uses complex instruction set computing (CISC), more features and ways to access memory compared to ARM
3. ARM uses reduced instruction set computing (RISC), more general registers for memory
4. X86 uses little-endian while ARM has the option to run on both little and big-endian modes
5. **What is the difference between sequential and parallel computation and identify the practical significance of each?**

Sequential – series of instructions, single processor, only one instruction at a time, longer

Parallel – instructions executed concurrently, multiple instructions at once, multi-core, faster

Parallel can be used in databases and virtual reality, for example. All modern computer systems support this and use it. Sequential is ok for tasks that are always the same and are fairly small.

1. **Identify the basic form of data and task parallelism in computational problems.**

Data parallelism – simultaneous execution of different functions on multiple cores across dataset elements

Task parallelism – simultaneous execution of same function on multiple cores across dataset elements

1. **Explain the differences between processes and threads.**

Processes – running program abstraction, programs do not share memory among each other

Threads – decomposed process into smaller, independent parts. Common memories shared

1. **What is OpenMP and what is OpenMP pragmas?**

OpenMP is a library language that supports multiprocessing programming on many platforms. OpenMP pragmas are compiler directives that generate threads.

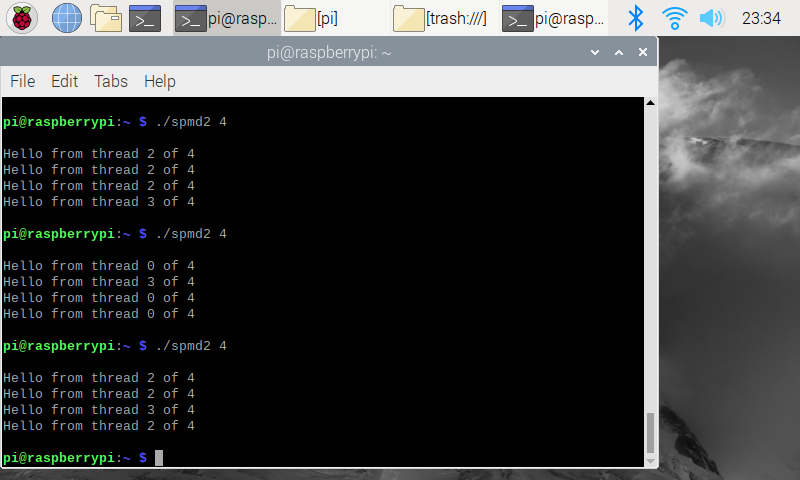
1. **What applications benefit from multi-core (list four)?**

Compilers, database servers, web servers, and most applications with thread-level parallelism benefit from multi-core processing.

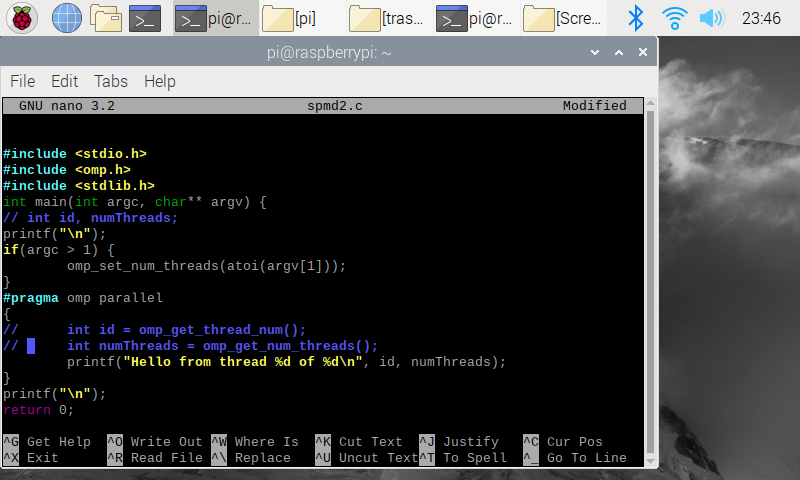
1. **Why Multicore? (why not single core, list four)**

Multi-core processors are easier to make compared to increasing single-core frequencies. Also, the future of application development is multithreaded, so multicores will be important to keep up with this trend in technology advancement. Parallelism is also preferred since it is quicker and can perform multiple operations at once. Multicore usage helps architects avoid technological obsolescence and improve maintainability.

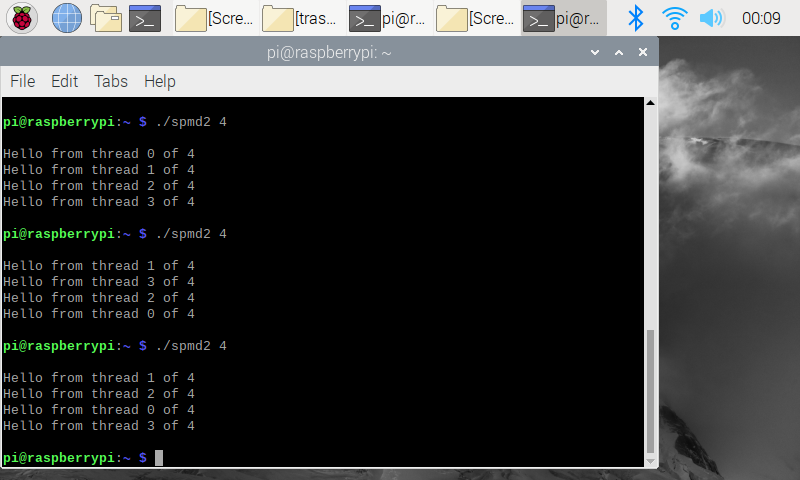
Parallel Programming Report

**Question 2.3 Faulty Code**

This is the output of the spmd2.c program, with the number 4 as the variable number of threads. The threads are not properly displayed or used since each thread did not have its own copy of the id and numThreads variables.

**Question 2.4.1 Fixed Code**

Changed code is highlighted in blue as a comment. After I took this screenshot, I uncommented the two integer assignments inside #pragma so each thread would have its own unique copy of the variable and could therefore could properly update the values with its own assigned thread number.



**Question 2.4.1 Fixed Code Output**

The new code produces the desired output of a unique thread number for each new line. No matter what order the threads finish, each is included in exactly one line of output.

ARM assembly Programming

Task 4

**Part 1 - Question 3 Program**

**A screenshot of a computer screen

Description automatically generated**The program is saved in second.s and then taken to the command line in order to complete the debug process and view info registers. While coding this program I also made myself more familiar with the ARM syntax and how to access values inside of the memory address.

**Part 1 – Question 3 Info Registers A screenshot of a computer screen

Description automatically generated**

Looking at the various registers, r1 has been updated with the correct value of 7 and r2 has stored the memory address of variable c.

A screenshot of a computer

Description automatically generated**Part 1 – Question 3 Variable Memory**

Shows setting the breakpoint and then running the program. While figuring out how to access the variable memory I figured out that you can search by the variable name in the same way that you can in x86 assembly language. Each variable has correctly been assigned its 32-bit integer value initialized in the .data.

A screenshot of a cell phone

Description automatically generated**Part 2 – Program Code**

The program is structured similarly to the previous program. Each val variable is assigned as a word so they fit the 32-bit size requirement. After I took this screenshot, I noticed that the integer immediate 9 was causing the program to not assemble. Adding a # to the front of it fixed that issue. The final ldr command was also an issue and I had to add square brackets to r2.

A screenshot of a computer

Description automatically generated**Part 2 – Program Info Registers**

After making the fixes, the program produces the desired output of 30 in the r2 register. I decided to remove the code that stored the final answer in r4 and instead simply leave it in r2, since all of the operations were already completed.

**Mezemir Gebre Individual Questions and Reports**

Parallel Programming Foundations

Task 3

* The raspberry PI B+ has the following components: CPU/Ram, Ethernet Controller, USB, HDMI, Display, Ethernet, camera.
* The Raspberry PI’s CPU is a quad-core or has 4 cores.
* The main differences between x86(CISC) and ARM (RISK) are the following: X86 processor by intel is a complex instruction set computing(CISC), which has a lot more and complex instruction sets to access the memory while Arm processors have reduced and less complicated instruction sets than the x86 processers. The intel x86 processors use the little-endian format to store date in the memory however the ARM processors use the BI-endian format since version 3. The other difference between them is for accessing the memory, The Arm processers use load/store and has more general registers unlike the x86 processors.
* The main difference between sequential and parallel computing is: parallel computing involves the concurrent or simultaneous execution of processes or threads at the same time while sequential computation involves consecutive and ordered execution of processes one after another.
* Data parallelism involves concurrent execution of the same task on each multiple computing cores and running task on different components of data, in contrast task parallelism involves distributing tasks simultaneously performed by threads across different processors.
* Process is the instance where a program is being executed in a computer(running). Thread is form of a process but involves dividing one process into smaller sequence of instructions that can be executed independently.
* Open MP is an application programming interface that supports shared memory programming and OpenMP programs are programs that helps the compiler to from a threaded code in order to have a concurrent execution of processes.
* Four applications that benefit from multi-core system are compliers, web servers, database servers, multimedia applications.
* Some of the reason to prefer multi-core over single-core are: it is hard and time consuming to increase a single-core’s clock frequencies, the applications that are being made now a days are multithreaded and in general parallelism is preferred for a computer architecture.

Parallel Programming Report

I followed the given instruction in the file “The parallel programming task A2” and got the following results. This program used an application programming interface called open MP so the CPU can decompose tasks and do parallel computation for efficiency. The first picture shows a C program that has been given, I just typed that in my Raspberry PI with a few modification as instructed. As showed in the second picture, I commented line 5 and added “int” in front of “id ”because, if we declare those variable outside the block that will be forked and run in parallel on separate threads, all threads share that variable’s memory and we won’t have a separate threads with a unique id. As the last screenshot shows, four lines printed from four different threads with a unique Id. That confirms to us that there was parallel computation going on and each of those threads have been assigned their own task and they are independent of each other. The reason for having four threads is because our raspberry PI B+ model has four cores, so the OS will schedule separate threads on each of those cores. A screenshot of a social media post

Description automatically generatedA screenshot of a social media post

Description automatically generatedA screenshot of a cell phone

Description automatically generated

ARM assembly Programming

**Part1**

The screenshot below is what I got after writing, assembling, debugging and analyzing the memory for the “second.s” program that has been given in the document (Arm\_Assembly\_programming\_A2 in icollege). As you can see, I put break on line 15 (address 0x10088) and analyzed the values in that address then it displayed those three hexadecimal values stored in that address.

**A screenshot of a cell phone

Description automatically generated**

**Part2**

The problem given here was to do the following arithmetic(val2+9+val3-val1) and store the result in the register. To do that, First I had to declare val1, val2 and val3 in the .data section as 32-bit integer variables and assigned their given value as you can see in the picture below. Since I was asked to verify the result in the memory and register, the next thing I did was to load the memory address of those variables into the registers, also their assigned value as well. The last thing I did was the arithmetic, added 9 to val2 and store it to r2 then I added 16 to r2 and store it in r2 again and subtract val1 from value 2 then add val1 to val2 and finally stored the result in r2 as you can see in the second picture, I got 30 in r2 register, which is the correct result for the given problem. The third picture is what I got after analyzing the memory for this program.

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Links to Slack, GitHub, YouTube

Slack Channel: georgia-state-cs 🡪 proj-assemblychefs <https://app.slack.com/client/TNCC77K3K/GTDJAQL05>

YouTube –

GitHub - <https://github.com/AssemblyChefs/CSC3210-AssemblyChefs/tree/master/Project%20A2%20Files>

Screenshots

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer screen

Description automatically generated**