**MANCHESTER METROPOLITAN UNIVERSITY**

**School of Computing, Mathematics & Digital Technology**

**ASSIGNMENT COVER SHEET**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| Unit: Computer Systems Fundamentals | Code: 6G4Z1102 |
| Assignment set by: | R.Cherry |
| Verified by: | D.Dancey |
| Moderated by: | D.Dancey |
| Assignment number: | 1CWK50: this (second part) is worth 25%  MCQ test in labs was worth 25% |
| Assignment title: | MIPs Assembler |
| Type: (GROUP/INDIVIDUAL) | INDIVIDUAL |
| Hand-in format and mechanism: | via Unit area on Moodle AND Coursework Receipting |
| Deadline: | As indicated on Moodle. |

Learning Outcomes Assessed:

Write, run and debug a program in assembler language.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

It is your responsibility to ensure that your work is complete and available for assessment by the date given on Moodle. If submitting via Moodle, you are advised to check your work after upload; and that all content is accessible. Do not alter after the deadline. You should make at least one full backup copy of your work.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Penalties for late hand-in: see Regulations for Undergraduate Programmes of Study: <http://www.mmu.ac.uk/academic/casqe/regulations/assessment.php>. The timeliness of submissions is strictly monitored and enforced.

Exceptional Factors affecting your performance: see Regulations for Undergraduate Programmes of Study : <http://www.mmu.ac.uk/academic/casqe/regulations/assessment/docs/ug-regs.pdf>

Plagiarism: Plagiarism is the unacknowledged representation of another person’s work, or use of their ideas, as one’s own. MMU takes care to detect plagiarism, employs plagiarism detection software, and imposes severe penalties, as outlined in the Student Handbook (<http://www.mmu.ac.uk/academic/casqe/regulations/docs/policies_regulations.pdf> and Regulations for Undergraduate Programmes (<http://www.mmu.ac.uk/academic/casqe/regulations/assessment.php> ). Bad referencing or submitting the wrong assignment may still be treated as plagiarism. If in doubt, seek advice from your tutor.

|  |  |
| --- | --- |
| Assessment Criteria: | Indicated in the attached assignment specification. |
| Formative Feedback: | One to one verbal during laboratory sessions |
| Summative Feedback format: | See Grid |
| Weighting: | This Assignment is weighted at 25% of the total unit assessment. |

Laboratory sessions will be used for formative feedback and support leading up to the final submission.

**MIPS Assembly Language Assignment**

The tasks you need to complete on this assignment are dependent on the last two digits of your student registration number.

If the last digit of your registration number is even (e.g 0, 2,4,4,6,8) then do Task 1.

If the last digit of your registration number is odd (e.g 1,3,5,7,9) then do Task 2.

**Task 1: (even registration number)**

Write a brief report, clearly divided into sections Task A, Task B and Task C which address the following:

**Task (code) A]** Using the MIPS architecture and assembly language create a fully commented programme that allows a user to input their name (one string) and their student registration number (an integer number) and then output both to screen.

|  |
| --- |
| **In your report**  Write 100 words on describing how the programme works and include a screen shot of both the input stage and the final output.  Copy and paste your code into a word document. |

**Task (code) B]** Using the MIPS architecture and assembly language create a fully commented programme that allows a user to input the last two digits of their student registration number then divides that number by 2 and outputs the result to screen.\*

Extend the programme to output to screen the number calculated above TEN times. Use a LOOP structure to accomplish this task.

|  |
| --- |
| **In your report**  Write a further 100 words on describing how the programme works and include a screen shot of both the input stage and the final output.  Copy and paste your code into a word document. |

**Task (code) C] This is a difficult task** that requires you to learn about and apply nested loops before making an attempt. It is worth approximately 20% of this assessment (5% of final unit mark). **It is worth attempting this part with a view to picking up extra marks even if you cannot complete the task.**

In the past one of the classic way of benchmarking a computer system (i.e determining how fast/good it is), was to get the computer to undertake large numbers of calculations and time how long it took. This was done by embedding a simple calculation in the heart of a nested loop and then increasing the number of loops within the nest

Recently Bitcoin mining (<https://en.bitcoin.it/wiki/Mining>) has made use of extremely fast processing capability which in many instances has been best won by using assembly language to produce very efficient code. In this assessment you will be using a programing structure which underpins the “mining” process. If successful with this task you will fully appreciate the extent of the processing problem involved in this technology.

Using the MIPS architecture and assembly language create a fully commented programme that allows a user to input ANY three digit number and then uses that number with as many nested do loops as are required to calculate the following:

Three digit number+ Three digit number = result

The “result” must NOT increment after each loop. i.e you simply repeat the same calculation time after time

Make sure that what every computer you are working on takes 30 seconds to complete all loops and calculations. This is done by progressively increasing the size of the three digit number.

At the end of the programme output ONCE the result of:

Three digit + Three digit number = result

If you input 999 (the largest three digit number possible) and the MIPs programme takes less than a second to run then add another nested loop. Continue to add loops until the time the programme takes to run can be altered between 5 and 30 seconds simply by adjusting the three-digit number inputted in by the user. i.e the number 999 should result in a run time of 30 seconds.

The number of nested loops required will depend on your coding efficiency and the computer system that you are working on.

Finally – it is possible to use the system time to monitor the start and end time of the loop. Implement this additional code and output to screen the exact time taken for every calculation. This is extremely hard to do!

|  |
| --- |
| **In your report**  Write a further 300 words describing how the programme works (or why it doesn’t – you will get marks for a critical reflection) and include a screen shot of both the input stage and the final output and Indicate the number of nested loops you need for your computer system to complete this task  Include a system description screenshot, which includes processor speed and system type. E.g for an Apple based computer:    Draw a flow diagram illustrating the logic behind your programme code.  Copy and paste your code into a word document. |

**Task 2: (odd registration number)**

Write a brief report, clearly divided into sections Task A, Task B and Task C which address the following:

**Task (code) A]** Using the MIPS architecture and assembly language create a fully commented programme that allows a user to input their name (one string) and their student registration number (an integer number) and then outputs both to screen.

|  |
| --- |
| **In your report**  Write 100 words on describing how the programme works and include a screen shot of both the input stage and the final output.  Copy and paste your code into a word document. |

**Task (code) B]** Using the MIPS architecture and assembly language create a fully commented programme that allows a user to input the last three digits of their student registration number adds one and divides that number by 2 and outputs the result to screen. If your number is 999 then use the number 997 as your working figure.

Extend the programme to output to screen the number calculated above Six times. Use a LOOP structure to accomplish this task.

|  |
| --- |
| **In your report**  Write a further 100 words on describing how the programme works and include a screen shot of both the input stage and the final output.  Copy and paste your code into a word document. |

**Task (code) C] This is a difficult task** that requires you to learn about nested loops a before making an attempt. It is worth approximately 20% of this assessment (5% of final unit mark). **It is worth attempting this part with a view to picking up extra marks even if you cannot complete the task.**

In the past one of the classic way of benchmarking a computer system (i.e determining how fast/good it is), was to get the computer to undertake large numbers of calculations and time how long it took. This was done by embedding a simple calculation in the heart of a nested loop and then increasing the number of loops within the nest

Recently Bitcoin mining (<https://en.bitcoin.it/wiki/Mining>) has made use of extremely fast processing capability which in many instances has been best won by using assembly language to produce very efficient code. In this assessment you will be using a programing structure which underpins the “mining” process. If successful with this task you will fully appreciate the extent of the processing problem involved in this technology.

Using the MIPS architecture and assembly language create a fully commented programme that allows a user to input ANY three digit number and then uses that number with as many nested do loops as are required to calculate the following:

(Three digit number) multiplied by (Three digit number) = result

The “result” must NOT increment after each loop. i.e you simply repeat the same calculation time after time

Make sure that what every computer you are working on takes 30 seconds to complete all loops and calculations. This is done by progressively increasing the size of the three digit number.

At the end of the programme output ONCE the result of:

Three digit \* Three digit number = result

Where “\*” indicates “multiplied by”.

If you input 999 (the largest three digit number possible) and the MIPs programme takes less than a second to run then add another nested loop. Continue to add loops until the time the programme takes to run can be altered between 5 and 30 seconds simply by adjusting the three-digit number inputted in by the user. i.e the number 999 should result in a run time of 30 seconds.

The number of nested loops required will depend on your coding efficiency and the computer system that you are working on.

Finally – it is possible to use the system time to monitor the start and end time of the loop. Implement this additional code and output to screen the exact time taken for every calculation. This is extremely hard to do!

|  |
| --- |
| **In your report**  Write a further 300 words describing how the programme works (or why it doesn’t – you will get marks for a critical reflection) and include a screen shot of both the input stage and the final output and Indicate the number of nested loops you need for your computer system to complete this task  Include a system description screenshot, which includes processor speed and system type. E.g for a Apple computer:    Draw a flow diagram illustrating the logic behind your programme code.  Copy and paste your code into a word document. |

-------------------------------------------------------------------------------------------------------------

**Submission of coursework.**

**You must submit in the following ways by the date advertised on Moodle:**

1] An electric submission of your document (in .pdf format) over the Unit area for CSF. Note that this also includes your code.

2] Copy and paste all code into a single text file. Indicate the sections. Then submit this over the Moodle unit area. This is a different place to the document submission and will make use of the University plagiarism detection system.

Note that all work will be checked for plagiarism – in particular, overt copying between students both current and past will be checked for.

**Marking Grid**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Learning Outcome** | Fail 0-40 | Pass 40-60 | Strong 60-80 | Excellent 80-100 |
| **Write, run and debug a program in assembler language.** | No working code for Sections  A or B  No/poor Documentation.  No Submission  Wrong Task Attempted as judged by registration number | Working code for Sections  A and B  Documentation for sections attempted  Loop structure not working or missing  Wrong Task Attempted as judged by registration number (-5%) | Working code for Sections  A,B,C  Documentation complete and articulate.  Simple loop structure in place for B  Part C shows some errors  Evidence of problem solving and autonomous learning  Wrong Task Attempted as judged by registration number (-5%) | Working code for Sections  A,B,C  Documentation succinct, concise, well expressed, objective with a focussed discourse  Part C working with a delivering a set of nested loops that control the time for calculation over a good range to 30 seconds.  Well-structured code with Good comments  Good critical reflection of why part C didn’t work (or did)  System time used to measure calculation duration  Wrong Task Attempted as judged by registration number (-5%) |