University of Nottingham Ningbo China

School of Computer Science

A Level 1 Module, Spring Semester 2022–2023

COMP1047: Systems and Architecture (AE1SYS)

Coursework (Assessment Weight: 50%)

Submission deadline: Friday 28th April 2023, 23:59:59 GMT+8

---

1 Synopsis

This coursework is about MIPS programming, implementation and testing, as well as conception of CPU design. You should answer all the **THREE (3)** questions below, before submitting them in Moodle and CS repository. For MIPS coding problems, you are advised to develop and test your code ONLY using the MIPS simulator adopted in the lab sessions in this course, which is QtSpim (v9.1.23). The total mark of this coursework is [**50 MARKS**].

2 Deliverable

Submit the following **FOUR (4)** *uncompressed* files to COMP1047 Moodle page, where you will find a submission portal to be opened later.

1. `COMP1047CW-'YourName'-'YourID'-Q1.s` for the code corresponding to Question 1. Example: `COMP1047CW-JaneDoe-20219999-Q1.s`.

2. `COMP1047CW-'YourName'-'YourID'-Q2.s` for the code corresponding to Question 2.

3. `COMP1047CW-'YourName'-'YourID'-Q3.txt`, in which you only need to write down your project name (AE1SYS-CW-YourID) in the csproject. Details can be found in Question 3's description.

4. `COMP1047CW-'YourName'-'YourID'-readme.txt` for whatever you would like to tell the evaluators. Word count limit is 500, including all components. This item is *optional*.

Unable to follow the file naming conventions would lead to mark deduction. Late submission rules apply, as indicated in the accompanying coursework issue sheet.

3 Plagiarism

You are gently reminded that we are at liberty to use plagiarism detection tools on your submission. Plagiarism will **absolutely** not be tolerated, and academic offenses will be dealt with in accordance with UNNC policy and as detailed in the student handbook. This means you may informally discuss the coursework with your classmates, but you must implement your own code and provide your own answers. **DO NOT** **copy and paste, or paraphrase from others.**

While being a promising personal assistant in many ways, **ChatGPT**, or other types of AI, is regarded as an academic offence in UNNC if leveraged for coursework purposes: "University policies have been updated so that it is beyond reasonable doubt that AI constitutes academic offence." – *ChatGPT Task and Finish Group: Guidance on AI and Assessment.* It is important to note that the university, as well as the industry, are investigating technologies to identify the ChatGPT-produced outputs. We reserve the right to back-check the coursework submissions in a future point, even beyond this module's period.

4 Assessment

1. This coursework has a total of 50 marks, which constitutes 50% of the total module marks. Individual marks are shown along with each question.

2. Detailed evaluation rubrics for Questions 1 is provided in Appendix 2. Evaluation rubrics for Questions 2 and 3 are provided in their problem descriptions, respectively. Note that, when evaluating your submissions, we may interview you regarding the details of the code, if needed.

3. For enquiries on doubts or issues encountered when doing your coursework, please use the Moodle discussion forum, or directly email Dr. Ying (for Q1) or Dr. Heng (for Q2 and Q3).

Question 1 [20 Marks]

Write a program in MIPS32 assembly language which takes three input arguments: register A will receive a character, register B will receive another character, and register C will receive the initial address of a string. Within the string, your program will replace any occurrence of the character stored in register B by the character stored in register A. Finally, output the replaced string in register C. For example:

- Input character in register A: `c`

- Input character in register B: `f`

- Input string in register C: `affording to use a silver fork`

- Output string in register C: `according to use a silver cork`

Detailed marking rubrics can be found in Appendix 2.

Question 2 [20 Marks]

Implement a MIPS version of the following C code, including the C function `strncat()` and `strstr()`, which is specified in the IEEE 1003.1-2017 Standard.

Your Task

Given the following C code snippet:

```c
1    #include <stdio.h>
2    #include <string.h>
3
4    int count_substr_n(const char*, const char*, int);
5
6    int main() {
7       const char s1[25] = STRING 1;
8       const char s2[25] = STRING 2;
9       int n = N;
10
11      int result = count_substr_n(s1, s2, n);
12      printf("%d", result);
13
14      return 0;
15   }
16
17   int count_substr_n(const char* p1, const char* p2, int n) {
18      char p2_new[25] = STRING 3;
19      int count = 0;
20      strncat(p2_new, p2, n);
21
22      char* temp = strstr(p1, p2_new);
23      while (temp != NULL) {
24          count++;
25          temp++;
26          temp = strstr(temp, p2_new);
27      }
28
29      return count;
30   }
```

Note: To execute the code snippet above, you may need to replace the RED colored content by syntactically correct the C code. For example:

```
7       char s1[10] = "Forever Shared Future";
8       char s2[10] = "revisit";
9       int n = 2;

18      char p2_new[25] = "a";
```

Then the program should output

```
1
```

Your are required to implement the MIPS assembly code that returns **EXACTLY** the same result as executing the code snippet above. You *should not* change any parts in the above code snippet, except the red colored content. No mark is awarded to this question if this requirement is not followed.

Input:
Once you execute your MIPS code by invoking the appropriate syscall, at the QtSpim console your program should intake a single string in the following format (Px indicates Parameter x):

```
P1:STRING 1;P2:STRING 2;P3:N;P4:STRING 3;
```

where the RED colored content should be replaced in the same way as the above C snippet. For example:

```
P1:"Forever Shared Future";P2:"revisit";P3:2;P4:"a";
```

Output:
After your code is executed based on the above input, at the QtSpim console, your program should output the corresponding result. For the example above, the output should be:

```
1
```

Exceptions:
What happens if the above C code returns an <u>run-time</u> error, after you supplied the input and run? Note: a run-time error is the type of abnormality that the compiler does not recognize, and can finish compiling; but when executing the code, the error will occur, such as dividing by zero, and index out-of-range.

If the C code reports a run-time error, your MIPS code should also report a run-time error, since your MIPS code should implement the C code. But in this coursework, your MIPS code does not have to output exactly the same error as the C code reports. At the simulator console, your MIPS program should only output the following message.

```
An error has occurred.
```

Again, you *do not* have to specify the error, but make sure your MIPS program generates *exactly* this error message, when the C code generates any run-time error message.

What happens if the above C code returns an compile-time error?
My test cases will not contain inputs that lead to a compile error (or warning). It is the job of the compiler to examine the compile error or warning, and the aim of this coursework is not writing a compiler for error checking. In other words, you can ignore the cases that cause compile-time errors and warnings. They will not appear in the test cases.

Resources

1. Use the built-in gcc compiler available in our cslinux server as the standard C code behavior reference. You can copy and paste the C code above into cslinux and run, to obtain the referential result. In other words, your program should output *exactly* the same 'count' value (except errors) as the C code outputs under gcc in cslinux.

2. To access cslinux and the gcc inside, refer back to your labs in the PGA module. A manual to access cslinux via X2Go is provided in the Moodle's coursework section. Figure 1 is a snapshot that hopefully helps you recall the usage of gcc on cslinux.

3. For remote students, if cslinux is unavailable, please use the OnilneGDB C compiler as the reference. For remote students, if you use OnlineGDB, please register with me by sending an email to `heng.yu@nottingham.edu.cn`, and attach a formal statement provided by the FoSE faculty as the verification of your remote status. Please do so before 7th April, 2023; otherwise I will mark your submission based on the results using cslinux.

4. Click here to refer to the formal definition of `strncat()` function, and here for `strstr()`.
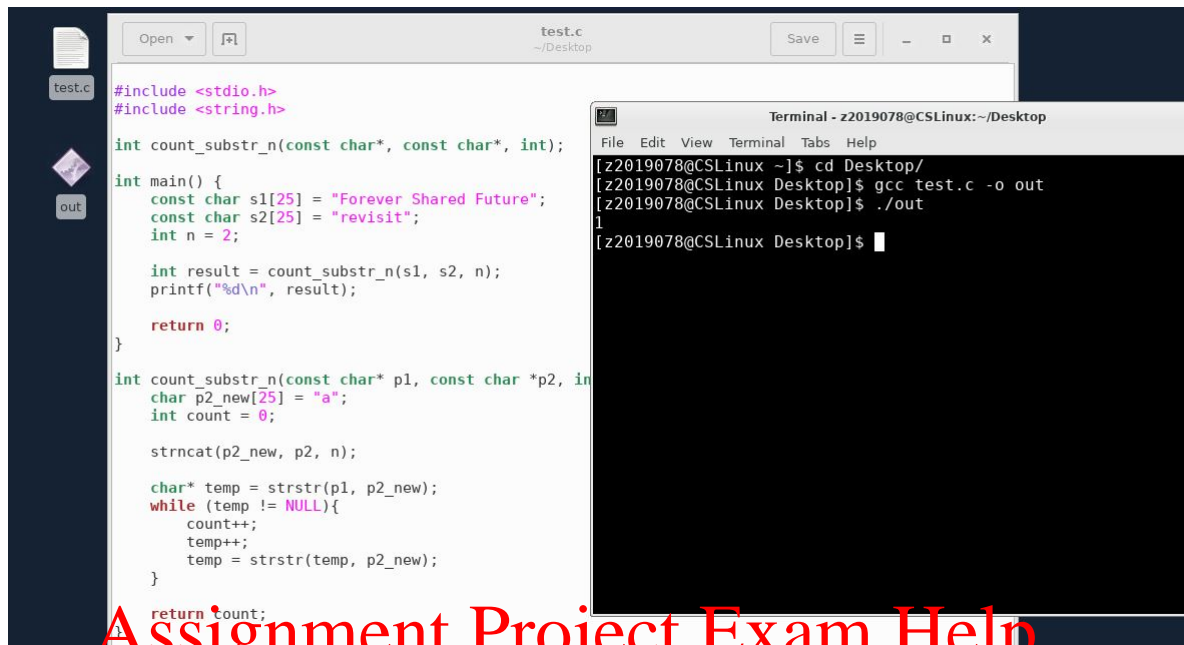
Figure 1: Demonstration of compiling and running C code on cslinux.

Evaluation

1. Out of the 20 marks for this question, TEN (10) <input, output> test cases will be employed to evaluate the functionality of your program. Example test case:

    <P1:"Forever Shared Future";P2:"revisit";P3:2;P4:"a", 1>.

    Each correct answer is rewarded 2 marks. Each incorrect answer is rewarded 0 mark. Your output answer should follow exactly the output requirement given in this question, otherwise it would be deemed incorrect.

2. Again, unless otherwise specified, we will use the gcc compiler currently available in cslinux and onlineGDB (only for remote students) as the C code behavior reference, to evaluate your program output.

Question 3 [10 Marks]

The Scenario

Due to the timing constraints, over the first 7 weeks, our COMP1047 module cannot fully cover all aspects of the computer architecture design. Now, assume that you are invited as a guest speaker to teach for one extra week, on the topic of computer architecture. When outlining your teaching content, you firstly identify 3 aspects that was not covered by the current COMP1047 materials. Then you choose one aspect that you are familiar with, and decide to teach it to the students.

Under this aspect of computer architecture, you organize several sub-topics that together constitutes the major domain knowledge of that aspect. They should be logically connected, but sufficiently different from each other. Then you start to prepare the details of each sub-topic. To give you a better concept of an 'aspect' and its 'sub-topics' – Example 1: in the aspect of ISA (Week 5), the sub-topics could be `R-type`, `I-type`, and `J-type` instructions. Example 2: in the aspect of pipeline hazards (Week 7), the sub-topics could be `structural hazards`, `data hazards`, and `control hazards`.

Your Task

Prepare a PPT with which you can talk for 5 minutes, specifically addressing the following questions:

1. Introduce the three (3) aspects that you find not covered, or not well covered, in this module. You do not need to dive into much technical details, but your introduction should clearly describe 'what they are', 'where are they in the big picture of computer architecture', 'what are their relationships with the architecture you have learnt in this module', etc.

   Then, pick one (1) of the above aspects talked, describe what are the sub-topics that you have prepared to teach. You don't have to introduce each sub-topic in detail, but only need to give their names, and describe 'what they are'. (5 marks)

2. Choose one (1) sub-topic above, and talk about it in extended technical depth. Moreover, describe the latest industry development or research achievement of this sub-topic. Besides depth, your description should also be correct and abundant, in order to let the audience quickly grab the key domain knowledge of this sub-topic. (5 marks)

Make an oral presentation using the PPT developed. Note that your audience is not the students to teach, but Dr. Heng as your assumed mentor to help you prepare for the

teaching. Also, it is important to note that, please do not add the content of Computer Networks as part of your answers, as this aspect will be taught in the last few weeks. Record a 5 minutes video of your presentation in `mp4` format, and submit **both your video and PPT** to our CS git server, namely csprojects.nottingham.edu.cn. To upload to csprojects, please:

    a. Create a new project named **AE1SYS-CW-YourID**, e.g., AE1SYS-CW-20411234.

    b. Upload your video and PPT into the project, and name them as **Video-YourID** and **PPT-YourID**, respectively. An illustration of your project page is shown below.

    c. You must add Dr. Heng as the project **Maintainer** in order for your project to be accessed. This can be done by `Project Information → Members → Invite member`, then input his email address (heng.yu@nottingham.edu.cn), and set him as the Maintainer in the 'Select a role' menu.

Figure 2: Demonstration of Q3 submission in csproject.

Evaluation

    Sub-tasks 1 & 2 will be evaluated individually, as their marks can be found right after the question. Basic marking criteria are `correctness`, `abundance`, and `depth`. For detailed rubrics, an example indicator is provided in the table below, to give you an direct impression

9

on how much effort should be corresponding to which level of grades.

Level	Example indicator
Fail	Nothing presented.
Poor	Provided the names only.
OK	Provided the names, with one or two sentences of description.
Good	In addition to OK level, provided extended elaboration trying to help the evaluator understand what is talked about.
Excellent	The information conveyed is correct and abundant. For sub-task 2, in-depth discussion and/or analysis is provided. High quality PPT. High quality usage of citations.

Besides the above, penalties of marking can be applied if

a. Video length is less than 4 minutes or more than 6 minutes.

b. Not showing your face in the video of presentation.

c. Not adding Dr. Heng as the maintainer to access your project.

d. Wrong project name, or file names uploaded, especially not showing your student ID.

e. Not uploaded a `COMP1047CW-'YourName'-'YourID'-Q3.txt` file into Moodle.

f. Any other factors that obviously lower down the submission quality, or make your project inaccessible.

Good luck! Enjoy MIPS Programming and Computer Architecting. Remember to submit your coursework on time.

Appendix 1: A previous year's example on Q2

Here we provide the question code and test cases for the previous year's COMP1047 module coursework.

```c
1    #include <stdio.h>
2    #include <string.h>
3
4    int main() {
5
6      char s1[10] = STRING 1;
7      char s2[10] = STRING 2;
8      int n = N;
9
10     strncpy(s1, s2, n);
11     printf("%s", s1);
12
13     return 0;
14   }
```

The test cases for the above code is listed below, for your reference.

No.	Input	Output
1	P1:"Together";P2:"Future";P3:2;	Fugether
2	P1:"Together";P2:"Future";P3:0;	Together
3	P1:"Together";P2:"Future";P3:-1;	An error has occurred.
4	P1:"T";P2:"F ture";P3:5;	F tur
5	P1:"T";P2:"Future";P3:23;	An error has occurred.
6	P1:"Together";P2:"Fu";P3:8;	Fu
7	P1:"NULL";P2:"Future";P3:5;	Futur"
8	P1:"Together";P2:"Future";P3:'a';	An error has occurred.
9	P1:"Together";P2:"Future";P3:0x04;	Fututher
10	P1:"Together";P2:"Future";P3:5-1* 3;	Fugether

Appendix 2: Rubrics for Question 1

Rubrics Q1	Weight (100)	Zero (0%)	Poor (20%)	Pass (40%)	Good (60%)	Excellent (80%)	Outstanding (100%)
Basic Function	40	No answer or all normal test cases failed	Iterative implementation, errors in normal test cases	Iterative implementation, no error in normal test cases, errors in special test cases	Iterative implementation, no error in normal test cases, no error in special test cases, other MIPS usage errors	Iterative implementation, no error in normal test cases, no error in special test cases, no MIPS usage errors	Excellent level + special considerations in data/control hazards (Encourage self-explorations)
Prompt	20	No prompt	Basic prompt to allow user's input	Advanced prompt to guide user's input to ensure normal input cases	Advanced prompt to guide user's input to ensure normal input cases and special input cases	Advanced prompt to guide user's input and warn consequences of special input. Present informative message to let user know of special input results	Advanced prompt to guide user's input and warn consequences of special input. Present informative message to let user know of special input results. Ensure service availability
Documentation	20	No comment. Very poor coding style	Few comments. Poor coding style	Insufficient comments. Good coding style	Comments on key instructions. Good coding style	Clear comments to explain the logic flow. Good coding style	Professional comments explaining program information, input/output, design considerations, etc. Good coding style
Input Test	20	No input test	Evidence of attempted input test, but failed	Attempted input test, only ensuring normal input	Identify one type of special input, reply with corresponding prompt	Identify and handle two types of special input, reply with corresponding prompt	Identify and handle more than two types of special input. Reply with excellent prompt for user's next input. Exhibit intelligence