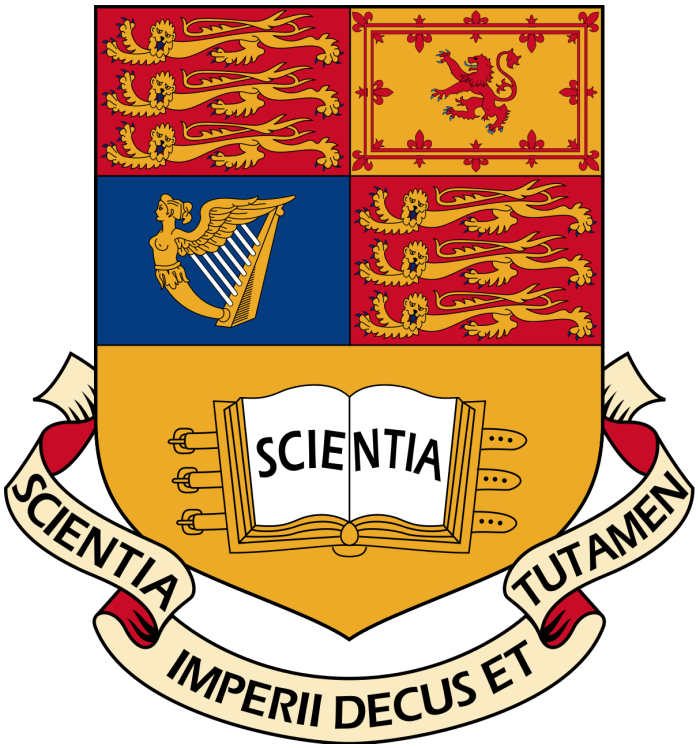
Imperial College London

Department of Electrical and Electronic Engineering

Time Management Log

Lex_on_them_haters

Tejas Dandawate
Aditya Gupta



1 Introduction

We had to keep track of how we allocated our time during the course of this project and this running log was kept to track our progress. In each meeting we discussed what we had done since the previous meeting as well as continuing to work on the project while together. We also established goals for each person which would ideally be accomplished by the next meeting. We also provided an estimate of how long we expect each last to take which is written beside each goal in brackets.

Each person would update their section of the log prior to each meeting commenting on the work they had completed since the last.

2 25th February 2020

2.1 Achieved During Meeting

We set up a meeting schedule, where we could discuss what occurred since the last meeting and what we wanted to achieve before the next one. We aimed to meet every 3-4 days in order to keep up to date with the work and give each other enough time to achieve the targets set out in each meeting.

2.2 Targets for Next Meeting

2.2.1 Aditya

Begin working on lexer.for operators and variables (20 minutes)

2.2.2 Tejas

Begin working on lexer.for C and Python keywords and variable (20 minutes)

2.2.3 Comments

We started by giving a light workload because we have to spend some time on our mathematics coursework.

3 28th February 2020

3.1 Achieved Prior to Meeting

3.1.1 Tejas

Finished working on lexer for C and Python keywords and variable

The work given did not take nearly as long as expected because we already had a lexer from the past labs that we could use and adapt.

3.1.2 Aditya

Finished working on C and python operators and tokens that are numbers.

The work took much less than 20 minutes because some features could be used from the lexer used in the preparatory labs. Other features not in the labs were found using the specification.

3.2 Achieved During Meeting:

During this meeting we joined together our lexer components, and made a make-file and file structure for the project.

MILESTONE ACHIEVED: Lexer Completed

3.3 Targets for Next Meeting:

3.3.1 Aditya

Independent research on how to implement the parser (1 hour)

3.3.2 Tejas

Independent research on how to implement the parser (20 minutes)

3.3.3 Comments

We still had to work on our maths coursework and also revise and complete the first language processors test, so we gave ourselves an open ended task of doing some research.

4 1st March 2020

4.1 Achieved Prior to Meeting

4.1.1 Tejas

Completed parser with Aditya

Using ANSI-C as a base for our parser we completed a basic parser.

4.1.2 Aditya

Completed parser with Tejas

We worked together to alter the open-source ANSI-C to work for our purpose.

4.2 Achieved During Meeting:

We simplified the ANSI-C parser to only include the constructs that were present in the specification for the python translator and the basic MIPS features.

MILESTONE ACHIEVED: Parser Completed for python translator

4.3 Targets for Next Meeting:

4.3.1 Aditya

Complete the variables and statements ASTs for python. Variables include implementation of both global and local variables. Statements includes the implementation of if else , while and return statements. (2 days)

4.3.2 Tejas

Complete the functions and expressions ASTs for python. (2 days)

4.3.3 Comments

Last week we underestimated the amount of work we could complete in the given time and so we gave ourselves slightly more work to complete before the next meeting. We wanted to finish python as quickly as possible to try and tackle MIPS generation which would be significantly harder.

5 5th March 2020

5.1 Achieved Prior to Meeting

5.1.1 Tejas

Completed the functions and expressions ASTs for python

Completing the python ASTs was challenging, but we still had access to the ASTs from our previous labs, and we were using the same file structure as well. This meant we could use the same print code to run through our tree and test if it was testing python correctly as well.

5.1.2 Aditya

Completed functions and expression ASTs for python

The time estimate for the python ASTs was quite accurate. It was a challenging yet achievable task due to the fact we could use the ASTs from the preparatory labs as a guide. The most challenging part was to ensure the printing of python was done in the correct manner as it was implemented recursively. Further, getting the indentation correct within each scope was also a demanding task.

5.2 Achieved During Meeting

Updated parser to include intermediate C features as well. This allowed us to be able to continue implementing new features individually.

MILESTONE ACHIEVED: Completed the python translator

5.3 Targets for Next Meeting

5.3.1 Aditya

Test the python translator using the tests provided on the master GitHub repository. (1 hour)

5.3.2 Tejas

Write test cases (including corner cases) to make sure that there were not any bugs in the translator (1.5 hours)

5.3.3 Comments

We wanted to make sure the python translator was functioning properly and was robust before moving onto working with MIPS. We also still have to work on our maths coursework before its fast approaching deadline, as well as the second language processors test. All this work was piling up so we wanted to keep the workload light.

6 9th March 2020

6.1 Achieved Prior to Meeting

6.1.1 Tejas

Tested our translator

I simply made tests myself and input them into our function called print_python. I found and fixed some minor bugs during this time, but the program was overall very much functional. This task was quite open ended, so I tried to keep to the 1.5 hours allocated for this task, but I think we overestimated how long it would take me, since I ended up running out of test cases before this time.

6.1.2 Aditya

Completed python testing for given tests

The time estimate was quite pessimistic as it was quite difficult to implement reading c code from a file and writing the outputted python to a file and took much closer to 5 hours to implement. I initially attempted to do

this in C++ but this turned out to be very time consuming and ultimately unsuccessful. A far better approach was to implement it using a bash script which tested the translator against each of the given test cases.

6.2 Achieved During Meeting

We firstly discussed any changes that we had made during our time debugging, to make sure we were both up to speed and we both had the master copy of our project. We also discussed time management, since we wanted to prioritise our time. Maths coursework seemed to be a more pressing issue to we delegated tasks, and postponed our next meeting to a slightly later date. We both agreed that we would implement the MIPS code using a stack based architecture as opposed to doing a purely register based approach.

6.3 Targets for Next Meeting

6.3.1 Aditya

Implement all C operators and function declarations and calls for MIPS. (4 days)

6.3.2 Tejas

Implement variable and conditionals ASTs for MIPS. (3 days)

6.3.3 Comments

We are giving ourselves a slightly larger amount to do, but also have compensated by giving ourselves a lot more time to complete it. This means that we can still stay on track and also keep up with our maths coursework.

7 18th March 2020

7.1 Achieved Prior to Meeting

7.1.1 Tejas

Completed variables and conditionals for MIPS

This was a slightly more involved task as although we could still use code given to use during labs as a starting point, we had to make sure it functioned with the rest of our code. Another thing that took some time to think about was how to implement the stack architecture, since we no longer had unlimited registers to work with. I decided to push all variables onto the stack, and pick them out when needed. This meant we required a new context class that would hold the location of each variable on the stack.

7.1.2 Aditya

Completed functions and operators for MIPS

This took slightly longer than the time estimated as the MIPS generation was a step up in difficulty from the python. The operators were not too difficult as they all employed a similar style. However, function invocations and definitions were more tricky due to the fact that I to ensure the frame pointer and stack pointer were in the correct locations by GCC convention. This is because the C driver files used to test the MIPS code would be compiled by the GCC compiler so our MIPS had to adhere to its conventions.

7.2 Achieved During Meeting

We decided that for the more complicated features it would be better for us to work more collaboratively, since the work we were doing was more complex and it was vital for both of us to understand what was happening at any given time. This meant we had to organise our time better and come up with a more rigorous meeting schedule for us to work together. During this meeting we also implemented for loops in MIPS.

MILESTONE ACHIEVED: Completed basic features for MIPS

7.3 Targets for Next Meeting

7.3.1 Aditya

Work on arrays for MIPS (5 hours)

7.3.2 Tejas

Implement break and continue statements (3 hours)

7.3.3 Comments

We will refrain from pushing anything onto the master git from now and instead work on a local git so that we don't push any faulty code, and we know what is happening as we edit our code. This also means that we won't have many confusing commits on the main git if we ever have to revert back.

8 22nd March 2020

8.1 Achieved Prior to Meeting

8.1.1 Tejas

Completed implementation of break and continue

Working on this took less time that expected, as the only thing that required extra thought was where to push labels onto a stack and where to pop them off, so that each break and continue would jump to the correct one.

8.1.2 Aditya

Implemented basic array structure

Arrays with constant and variable indexing was implemented but was not fully functional. The estimated time of completion was severely underestimated as there were still many bugs with arrays and not all possible cases for array access that still needed to be considered.

8.2 Achieved During Meeting

The arrays were not fully functional (as shown by the test bench) and so we had to read through the generated MIPS files to find where the error was. We fixed the arrays enough to pass constant and variable indexing, but there was still expression indexing and many other corner cases that we still couldn't compile.

8.3 Targets for Next Meeting

8.3.1 Aditya

Expression indexing for arrays. (5 hours)

8.3.2 Tejas

Implement switch statements. (3 hours)

8.3.3 Comments

We want to carry on implementing arrays to be more functional, but it would not be beneficial for us both to work on the same thing, so the other person will work on switch statements instead.

9 25th March 2020

9.1 Achieved Prior to Meeting

9.1.1 Tejas

Completed basic implementation of switch statement

I was able to implement the basic switch statements, but this took longer than expected, since it was particularly hard making the classes for this functionality. There were also a lot of cases to be accounted for even for the simple implementation of a switch, and so I decided that it would not be worth the time to try to make more complex cases work too.

9.1.2 Aditya

Completed expression indexing for simple cases

Expression indexing for basic expressions was functional but there was still much more that could be tested that I was unable to implement in the time between this meeting and the last. Arrays is the hardest feature encountered thus far due to the sheer number of different possible cases that must be considered and implemented.

9.2 Achieved During Meeting

During this meeting we were testing other edge cases for switch and arrays, and found that we did not pass for some cases of both. We could not have nested switches for example.

SCOPE OF DELIVERABLE REDUCED: We tried to implement these but we were unable to and decided the time it would take to do so would not be well spent, as it may not be worth the time.

9.3 Targets for Next Meeting

9.3.1 Aditya

Implement global variables for MIPS (4 hours)

9.3.2 Tejas

Research how enums work in C, and how it would be possible to implement them in MIPS. (2 hours)

9.3.3 Comments

We want to move on and implement more intermediate features, and we decided to meet more often in the future since the deadline was fast approaching.

10 26th March 2020

10.1 Achieved Prior to Meeting

10.1.1 Tejas

Completed research for enum

This was a completely new concept for us, even in C, and so we thought it would take more time to research. However this was a gross over-approximation, and the implementation of enum was not that complicated, given the test-benches provided. It was clear that we needed to implement global variables for enums to be implemented (as we were going to implement each enum as a list of global declarations).

10.1.2 Aditya

Completed basic global variable operations

Global variables were not too difficult to implement but they had still not be implemented rigorously. The time estimate was enough to implement them on a basic level but more time and thought would be needed to cover all the possible cases involving global variables.

10.2 Achieved During Meeting

Due to unforeseen circumstances (Covid-19), we were no longer able to meet in person and so we had to Skype our meeting instead. During it, we were able to implement enums using the structure already implemented for global variables.

10.3 Targets for Next Meeting

10.3.1 Aditya

Research why our recursive tests were failing. (2 hours)

10.3.2 Tejas

Research why our recursive tests were failing. (2 hours)

10.3.3 Comments

We want to complete the implementation of intermediate features, and the only one we were missing was recursive function calls. In order to implement this we decided to independently find out why the function calls were failing.

11 28th March 2020

11.1 Achieved Prior to Meeting

11.1.1 Tejas

Fixed function parameter storing/loading

During my research it became apparent that the way we stored parameters was inherently flawed, as it did not work for when we had an input parameter that featured operators (e.g. n+1). I therefore decided to fix this, but that meant that the time I had initially been allotted for this task was not nearly enough, and was in fact grossly underestimated.

11.1.2 Aditya

Fixed function calls

I found that after a function call the stack pointer, frame pointer and the return address were not being restored correctly and therefore needed to be fixed.

11.2 Achieved During Meeting

We updated each other on the fixes we had made relating to function calls and combined our locally edited files. We then decided to test recursive function calls and found that they were now functional.

MILESTONE ACHIEVED: Completed all intermediate MIPS features to a basic level.

11.3 Targets for Next Meeting

11.3.1 Aditya

Debug intermediate features and implement them more rigorously.(4 hours)

11.3.2 Tejas

Debug intermediate features and implement them more rigorously.(4 hours)

11.3.3 Comments

Now that we have implemented all the intermediate features to a certain degree, we want to make sure that they are robust, and so we will both run our external test-benches to find and fix any errors we may encounter.

12 29th March 2020

12.1 Achieved Prior to Meeting

12.1.1 Tejas

Fixed array access

While doing further research it became apparent that arrays still did not work entirely as intended, and so I made some changes to make sure they worked in a more robust manner. It became apparent though that this was an open ended problem and so I decided not to go over my allotted time for this task

12.1.2 Aditya

Debugged Unary Operators

While conducting some tests, I realised that our unary operators were not quite correct so I proceeded to correct these. I also found that our while loops were incorrect and so fixed them as well.

12.2 Achieved During Meeting

During the meeting we realised that due to some of the changes we had made earlier, our python was not completely functional. With our arrays, it was obvious that we had not accounted for all cases, but we decided it would not be worth spending more time for the few extra marks we may receive. We also managed to make functions that take more than four parameters work but we still need to test its functionality for completeness. After some testing, that our method did not exactly follow GCC convention, but we decided it was not worth our time to change this.

SCOPE OF DELIVERABLE REDUCED: We did not want to implement functions with over four parameters to comply with GCC convention.

12.3 Targets for Next Meeting

12.3.1 Aditya

N/A

12.3.2 Tejas

N/A

12.3.3 Comments

We have decided that as the deadline is almost upon us that we will work all day together tomorrow and that this will be the last day we spend on the project as we do not want a last minute panic to submit.

13 30th March 2020

13.1 Achieved Prior to Meeting

13.1.1 Tejas

N/A

We had no targets set as there was no time to do extra work between meetings.

13.1.2 Aditya

N/A

We had no targets set as there was no time to do extra work between meetings.

13.2 Achieved During Meeting

We decided to implement a final few things for arrays but we still had a few cases that we had not implemented.

SCOPE OF DELIVERABLE REDUCED: We tried to implement these but we were unable to and decided the time it would take to do so would not be well spent, as it may not be worth the time.

We also did some cleaning up of our code which allowed us to debug a few more basic features that were not robustly implemented. We wanted to attempt some of the advanced features for our personal satisfaction. We decided to implement external function declarations. After a few hours of work on the task, we were finally able to implement it.

MILESTONE ACHIEVED: Completed all advanced features we wanted to implement, and decided that the project was ready for submission.

13.3 Targets for Next Meeting

13.3.1 Aditya

N/A

13.3.2 Tejas

N/A

13.3.3 Comments

We have decided that we are happy with the project, and have decided that it is ready for submission.