University College London

Department of Electronic and Electrical Engineering

# Performance characterisation of 8-bit RISC and OISC architectures

| *Author:* | *Supervisor:* | *Second Assessor:* |
|---|---|---|
| Mindaugas Jarmolovicius | Prof. Robert Killey | Dr. Ed Romans |
| zceemja@ucl.ac.uk | r.killey@ucl.ac.uk | e.romans@ucl.ac.uk |

**A BEng Project Interim Report**

December 13, 2019

## 1 Abstract

This is abstract.

## 2 Introduction

Since 80s there been a raise of many processor architectures that try to fulfil specific performance and power application constraints. One of noticeable cases is ARM RISC (Reduced Instruction Set Computer) architecture being used in mobile devices instead of more popular and robust x86 CISC (Complex Instruction Set Computer) architecture in favour of simplicity, cost and lower power consumption [3, 2]. Its been shown that in low power applications such as IoTs (Internet of Things), OISC[1] (One Instruction Set Computer) implementation can be superior in power and data throughput comparing to traditional RISC architectures [5, 1]. This project proposes to compare two novel RISC and OISC architectures and compare their performance, design complexity and efficiency.

Project is split into 3 main objectives:
• Design and build a RISC based processor. As it is aimed for low power and performance applications this will be 8bit data size processor with 4 general purpose registers.
• Design and build an OISC based processor. There are multiple different implementations such as `SUBLEQ` or one proposed in chapter 3.1.
• Design a fair benchmark that both processors could execute. Benchmark may include different algorithms that are commonly used in controllers, IoT devices or similar low power microprocessor applications.

Following chapters will describe the estimated project outcome, project schedule and work done so far.

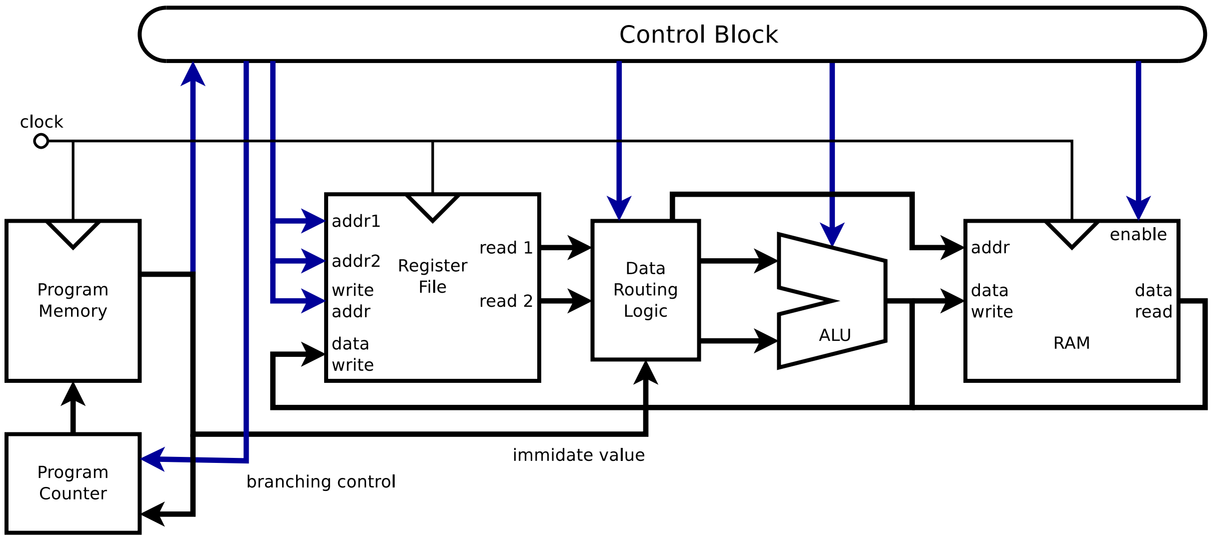
## 3 The Work Performed to Date

### 3.1 Supporting Theory

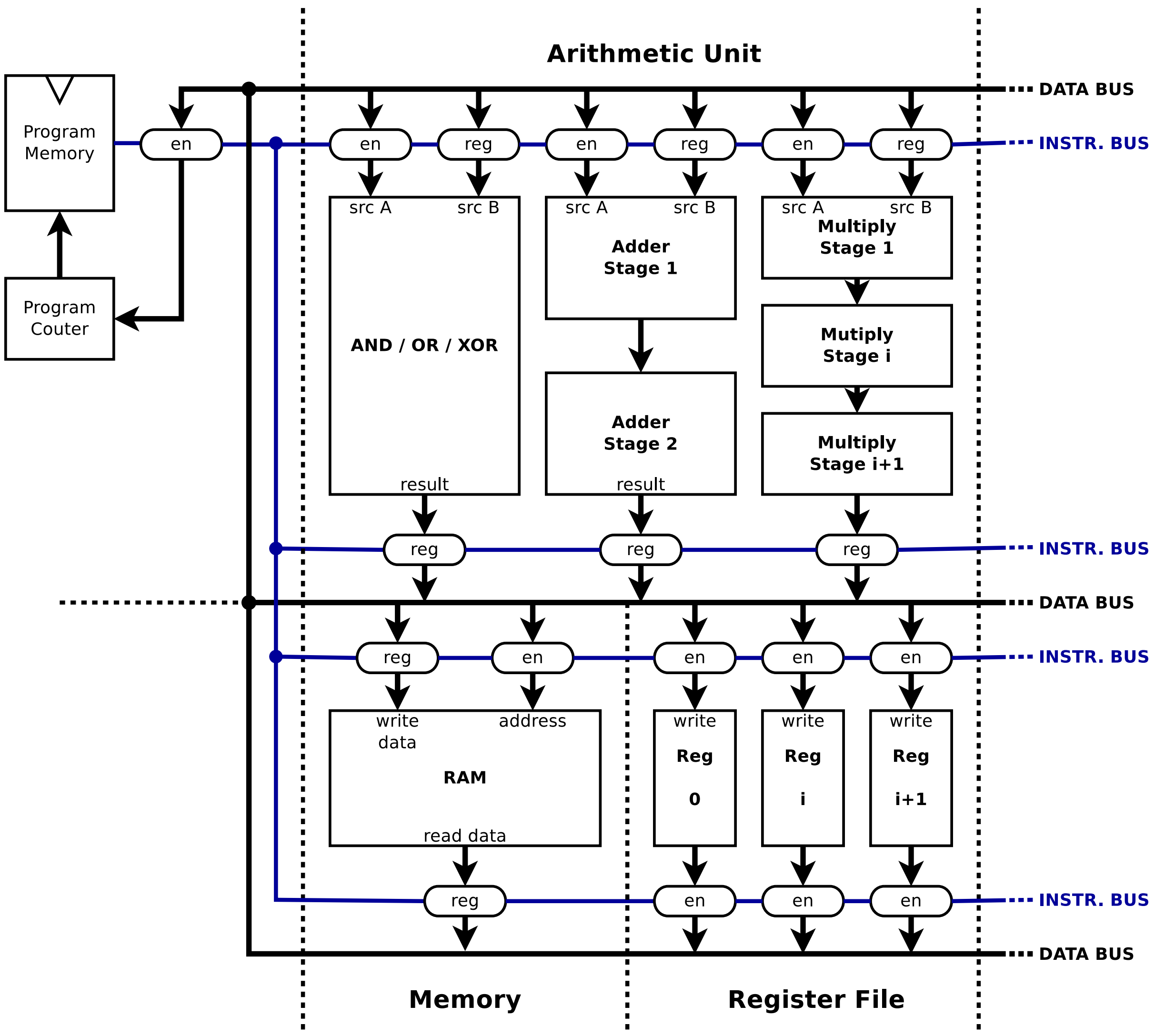
This section explains the theory and predictions of RISC and OISC architectures.

Figure 3.1.1 represents simplified diagrams of RISC and OISC architectures. In RISC architecture, program data travels from program memory to control block where instruction is decoded and further decided where what data is directed. Such structure requires complicated control block and additional data routing blocks. In order to increase performance of such processor one would need to add pipelining or add multiple cores. Both methods bring big disadvantages - multicore processor requires software adjustments and each core doubles the control and datapath substantially increasing die area; pipelinig allow operation at higher frequencies however it brings design complications such as complicated hazard prevention logic and instruction lookup. Simplicity of OISC architecture overcomes these disadvantages by following: Pipelining can be done by individual blocks and programmibly waiting for results, multicore can be simulated by adding more data and instruction buses, hazards can be prevented with software or/and integrated into address registers. Furthermore, ALU and any other processor component can be divided by adding different address registers thus allowing to utilise multiple components at the same time given that multiple data buses are used.

[1]Also known as URISC (Ultimate Reduced Instruction Set Computer)



(a) RISC microarchitecture diagram



(b) Single data bus OISC microarchitecture diagram

Figure 3.1.1: Simplified diagrams of both architectures. Blue lines indicate control/instruction buses and black - data buses

## 3.2 Project Scheduling

As it can be seen in table 1 below, project is mainly split into Term 1 which is dedicated for RISC and Term 2 which is dedicated for OISC implementation. Approximately 3 weeks are left until final report to have enough spare time to finish all tests and complete poster & report itself. It is also expected to have a lot of coursework around this time.

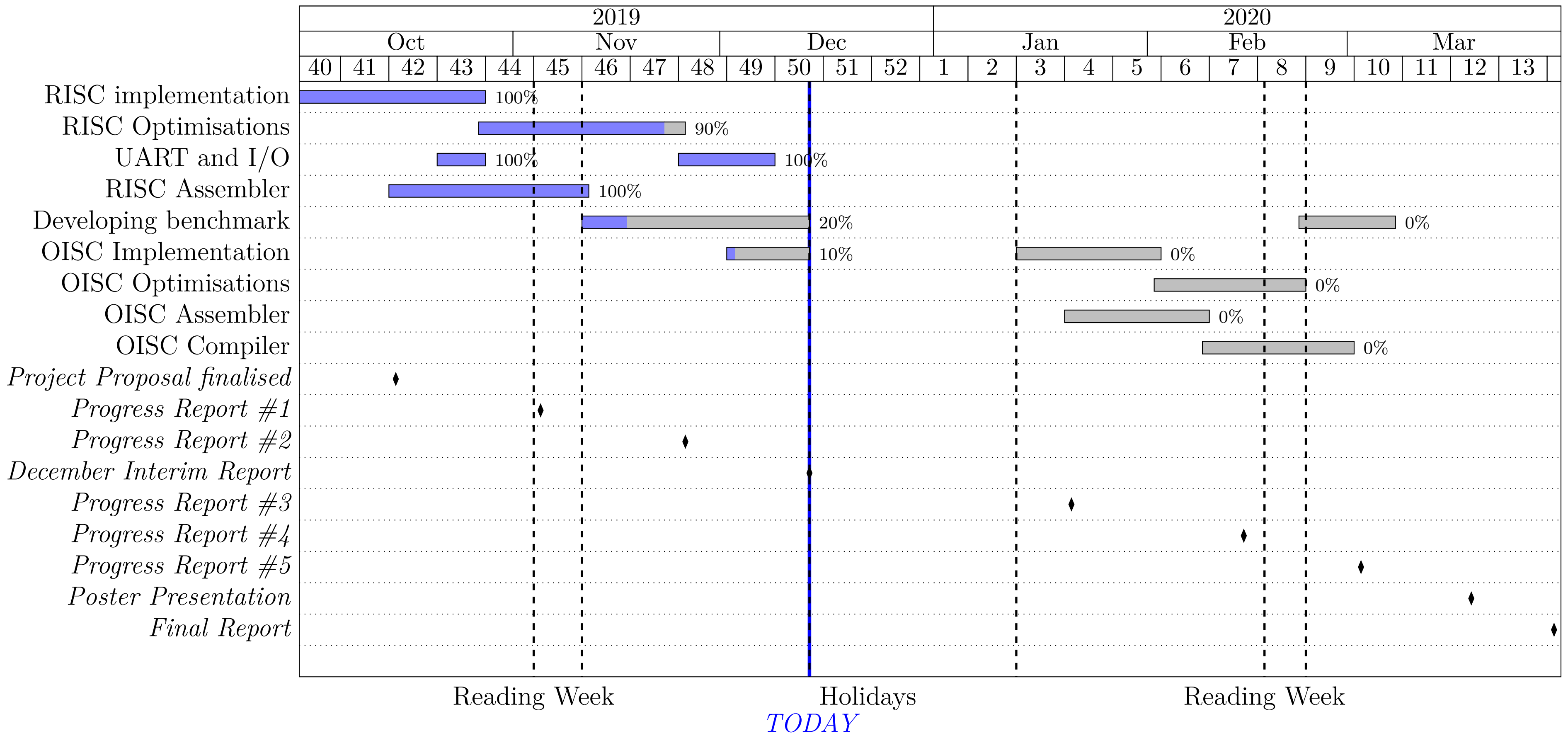


Table 1: Project schedule Grantt chart

## 3.3 Progress to Date

Use as many sections as you need to in order to discuss your progress so far. For example, you can use a different section to discuss each bit of the system to be constructed or designed.Include difficulties and issues impeding progress.

### 3.3.1 Memory

Initial plan was to use 32M-Word 16bit SDRAM chip located at FPGA board. After successfully simulating most of processor functions, next step was to synthesise and run it on FPGA which brought 2 problems: (1) Uploading program into ROM is not simple as generic Verilog unpacked register array cannot be initialised from file. Solution to this problem was solved by using FPGA built-in M9K memory that allows flexible RAM/ROM configuration. In addition, M9K memory can be read from/written to via JTAG connection without affecting FPGA operation which enabled quick method to upload programs without need to resynthesising processor code.

(2) Timing of SDRAM memory controller runs at much higher frequency than processor (at 100MHz versus 1MHz), initial implementation of interface between them was multiple 1-word length FIFO registers which caused memory read operation take 2 processor cycles. 3 possible solutions were considered - suspend processor clock while memory data is read which would harm performance; remove FIFO registers and rely on SDRAM clock being much greater than processor clock which is not ideal if higher processor clocks are intended to be used; use M9K. Last option was chosen due to ease of implementation and ability to read RAM content via JTAG for debugging purposes.

### 3.3.2 Instructions

Table 2 below represents RISC processor instructions that been implemented so far.

| Instr. | Description | Completed |
|---|---|---|
| | *2 register instructions* | |
| MOVE | Copy intimidate or register | x |
| ADD | Arithmetical addition | x |
| SUB | Arithmetical subtraction | x |
| AND | Logical AND | x |
| OR | Logical OR | x |
| XOR | Logical XOR | x |
| MUL | Arithmetical multiplication | x |
| DIV | Arithmetical division (inc. modulus) | x |
| | *1 register instructions* | |
| CI0 | Replace intimidate value byte 0 for next instruction | x |
| CI1 | Replace intimidate value byte 1 for next instruction | x |
| CI2 | Replace intimidate value byte 2 for next instruction | x |
| SLL | Shift left logical | |
| SRL | Shift right logical | |
| SRA | Shift right arithmetical | |
| LWHI | Load word (high byte) | x |
| SWHI | Store word (high byte, reg. only) | x |
| LWLO | Load word (low byte) | x |
| SWLO | Store word (low byte, stores high byte reg.) | x |
| INC | Increase by 1 | x |
| DEC | Decrease by 1 | x |
| GETAH | Get ALU high byte reg. (only for MUL & DIV) | x |
| GETIF | Get interrupt flags | x |
| PUSH | Push to stack | x |
| POP | Pop from stack | x |
| COM | Send/Receive to/from com. block | x |
| ADDI | Arithmetical addition with intimidate | x |
| SUBI | Arithmetical subtraction with intimidate | x |
| ANDI | Logical AND with intimidate | x |
| ORI | Logical OR with intimidate | x |
| XORI | Logical XOR with intimidate | x |
| BEQ | Branch on equal | x |
| BGT | Branch on greater than | x |
| BGE | Branch on greater equal than | x |
| BZ | Branch on zero | x |
| | *0 register instructions* | |
| CALL | Call function, put return to stack | x |
| RET | Return from function | x |
| JUMP | Jump to address | x |
| RJUMP | Relative jump | |
| RETI | Return from interrupt | x |
| INTRE | Set interrupt entry pointer | x |
| CLC | Clear ALU carry-in | |

| | | |
|---|---|---|
| SETC | Set ALU carry-in | |
| CLS | Clear ALU sign | |
| SETS | Set ALU sign | |
| SSETS | Enable ALU sign | |
| CLN | Clear ALU negative | |
| SETN | Set ALU negative | |
| SSETN | Enable ALU negative | |

Table 2: Instruction set for RISC processor

### 3.3.3 Program Counter

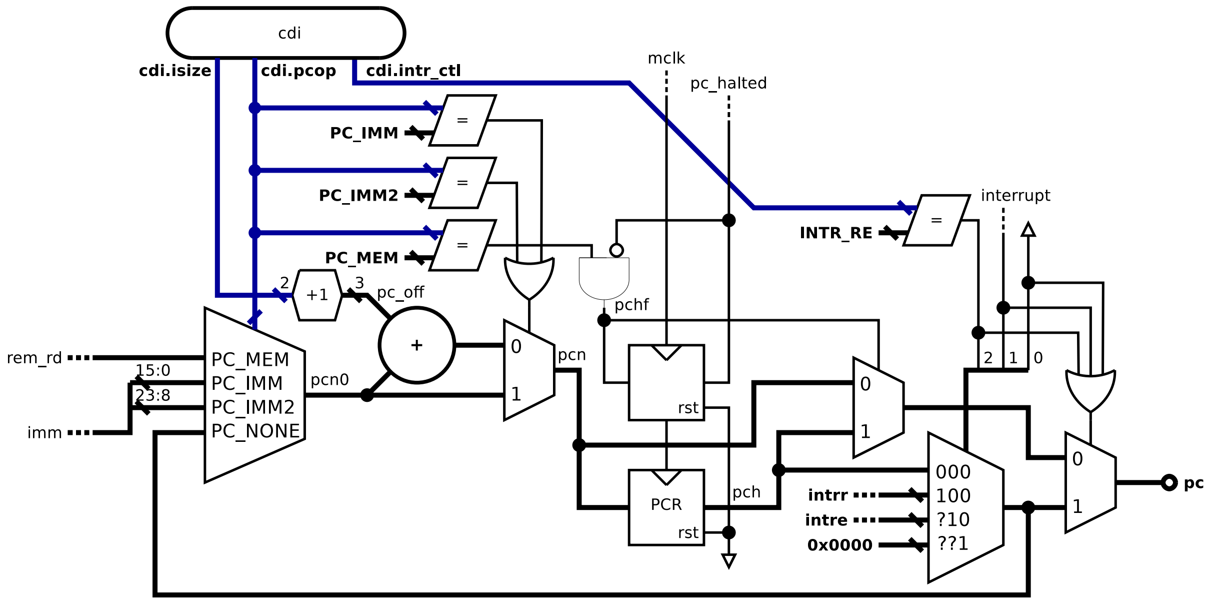


Figure 3.3.1: Digital diagram of RISC8 program counter

### 3.3.4 Stack Pointer

# 4 Summary of Difficulties and Issues

This chapter focuses on any difficulties and issues that are hindering project from moving forward.

## 4.1 List of Difficulties

List of difficulties currently encounter:

- Benchmark
- Assembler/Compiler

#### 4.1.1 Benchmark

One of the difficulties is to design an appropriate benchmark that could test scenarios used in actual such processor applications. Other issue comes with writing benchmark itself, for instance benchmark test to finding prime number may have used many different algorithms where one of the fastest and used in actual industry may be "Sieve of Atkin" algorithm [4], however is it fairly complicated and time consuming especially when written in assembly.

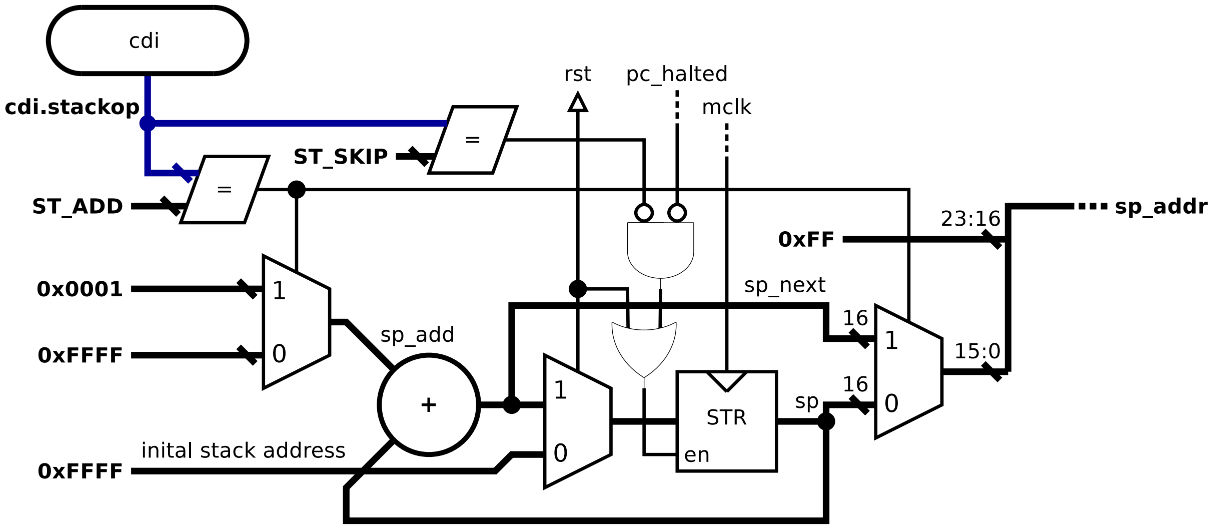


Figure 3.3.2: Digital diagram of RISC8 program counter

### 4.1.2 Assembler/Compiler

In order to write more complicated code for benchmark, a proper compiler is required. Current assembler support definitions and labels, however not macros or imports from other files, nor linked libraries. Ideally, a common language compiler needs to be developed so that benchmark programs could be imported without need to completely rewrite them in assembly. However such compiler might take more time to develop than writing benchmarks in assembly. Therefore it needs to be decided which option to do.

## 4.2 Failure Assessment

This section describes likely possibilities of project failures:

As of current schedule OISC processor will be implemented in Term 2, however due to personal schedule project will be given about twice less time than in Term 1 which may result in not finishing OISC processor or developing all benchmarks for it on time. Mitigation for this is to closely follow schedule and adjust OISC design in such way as it would take less time to implement benchmark.

Another possibility may be FPGA failure which would delay testing and benchmarking processors.

## 4.3 Updated Safety Risk Assessment

There are no changes to Safety Risk Assessment.

# 5 Appendix A: Safety Risk Assessment

RiskNet report is appended at the end of this document.

# 6 Appendix B: Computer Code

Append any code you may need here. Reference it in the text as "Appendix B, code snippet #"; for example,"Finda sample of the code usedfor this experiment in Appendix B, code snippet 2".

## 7 References

## References

[1] Tanvir Ahmed et al. “Synthesizable-from-C Embedded Processor Based on MIPS-ISA and OISC”. In: *2015 IEEE 13th International Conference on Embedded and Ubiquitous Computing* (2015). DOI: 10.1109/euc.2015.23.

[2] E. Blem, J. Menon, and K. Sankaralingam. “Power struggles: Revisiting the RISC vs. CISC debate on contemporary ARM and x86 architectures”. In: *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)* (2013). DOI: 10.1109/hpca.2013.6522302.

[3] T. Jamil. “RISC versus CISC”. In: *IEEE Potentials* 14.3 (1995), pp. 13–16. DOI: 10.1109/45.464688.

[4] François Morain. “Atkin’s Test: News from the Front”. In: *Lecture Notes in Computer Science* (1989), pp. 626–635. DOI: 10.1007/3-540-46885-4_59.

[5] Minato Yokota, Kaoru Saso, and Yuko Hara-Azumi. “One-instruction set computer-based multicore processors for energy-efficient streaming data processing”. In: *Proceedings of the 28th International Symposium on Rapid System Prototyping Shortening the Path from Specification to Prototype - RSP ’17* (2017). DOI: 10.1145/3130265.3130318.

# Risk Assessment

## Summary

**Reference:** RA030726/1 — **Sign-off Status:** Authorised

| | | | |
|---|---|---|---|
| **Date Created:** | 09/10/2019 | **Confidential?** | No |

| | |
|---|---|
| **Assessment Title:** | 3rd year project: Performance characterisation of 8-bit RISC and OISC architectures |
| **Assessment Outline:** | New activity |

| **Area Responsible (for management of risks)** | | **Location of Risks** | On-Site |
|---|---|---|---|
| **Division, School, Faculty, Institute:** | Faculty of Engineering Science | **Building:** | Roberts Building |
| **Department:** | Dept of Electronic & Electrical Eng | **Area:** | Ground and Above |
| **Group/Unit:** | All Groups/Units | **Sub Area:** | Laboratory |

| | |
|---|---|
| **Further Location Information:** | Roberts building Rooms 704, 905. Also working from home. |

| | | | |
|---|---|---|---|
| **Assessment Start Date:** | 09/10/2019 | **Review or End Date:** | 31/03/2020 |

**Relevant Attachments:**

**Description of attachments:**

**Location of non-electronic documents:**

| | |
|---|---|
| **Assessor(s):** | Jarmolovicius, Min |
| **Approver(s):** | ROBERT KILLEY |
| **Signed Off:** | ROBERT KILLEY (09/10/2019 12:56) |
| **Distribution List:** | Gerald McBrearty (g.mcbrearty@ucl.ac.uk) - 09/10/2019<br>ANDREW MOSS (andrew.moss@ucl.ac.uk) - 09/10/2019 |

**PEOPLE AT RISK (from the Activities covered by this Risk Assessment)**

| CATEGORY |
|---|
| Undergraduates |

Risk Assessment

## Activities, Hazards, Controls

**Reference:** **RA030726/1** **Sign-off Status:** **Authorised**

### 1. Working in a lab

**Description of Activity:**

**Hazard 1. Using computer**

| | |
|---|---|
| RSI Eye strain Bad posture | **Existing Control Measures**<br>Properly use mouse/keyboard Make constant breaks Make sure sit properly |

**Hazard 2. Testing FPGA**

| | |
|---|---|
| Burns from hot wires/chips due to short-circuits | **Existing Control Measures**<br>Use current limit on power supplies |

**Risk Level**

With Existing Controls:

| | |
|---|---|
| Risk Level | **A - Very Low / Trivial** |