

Contents

[**Introduction**](#_heading=h.30j0zll) **1**

[**Logisim CPU Simulation**](#_heading=h.7ytfx0es7kwm) **2**

[**Testing the CPU**](#_heading=h.2et92p0) **2**

[Data loaded](#_heading=h.tyjcwt) 2

[End of Program](#_heading=h.usnszwwo8v56) 3

[Amendments to the CPU](#_heading=h.jzwydi1b6f14) 4

[Implementing a reset button](#_heading=h.2s8eyo1) 4

[Data loaded](#_heading=h.eip1v4ejggxt) 4

[After reset](#_heading=h.yuruy881jw8x) 5

[Adding the halt instruction](#_heading=h.17dp8vu) 6

[Adding functionality to the ALU](#_heading=h.3rdcrjn) 6

[Multiplication](#_heading=h.tx40wukl6j81) 6

[Inversion](#_heading=h.klgntqkrizhe) 11

[Final ALU](#_heading=h.9r6f2ksaflir) 13

[Adding functionality for branch instructions](#_heading=h.26in1rg) 13

[Branch Always](#_heading=h.j9x2mkcak2ia) 13

[Branch Conditionally](#_heading=h.u95xmrw304we) 14

[Incorporating Input and Output devices](#_heading=h.lnxbz9) 17

[Making it easy](#_heading=h.z4hegy2on530) 18

[**Breaking it down**](#_heading=h.pvkhbsz6dsl4) **19**

[Instruction layout](#_heading=h.57x0bph99c8k) 19

[Instruction Decoding Unit (IDU) instruction list](#_heading=h.54uuqmxo23n7) 19

[ALU Instruction list](#_heading=h.e8uj4bv84d63) 19

[The Instructions (and the memory locations)](#_heading=h.nuj2iqevcz2z) 19

[Buttons](#_heading=h.96xqesvpa5t2) 20

[Loading a program and running it](#_heading=h.hpnwwm7m2dhg) 21

[**MIPS Assembly Language Programming**](#_heading=h.35nkun2) **22**

[**Testing the existing program**](#_heading=h.1ksv4uv) **22**

[**Amendments to the program**](#_heading=h.44sinio) **23**

[Improvements to messages output](#_heading=h.2jxsxqh) 23

[Using a subroutine](#_heading=h.z337ya) 24

[Adding functionality](#_heading=h.3j2qqm3) 25

[Adding a loop](#_heading=h.1y810tw) 27

[Using an array](#_heading=h.4i7ojhp) 28

[How arrays work](#_heading=h.9mq2rqe7lzam) 30

[Initialization](#_heading=h.84rzmfmjlcz1) 30

[Saving](#_heading=h.scemsjb7k55m) 30

[Reading](#_heading=h.bid72iaqsztf) 30

[**Discussion**](#_heading=h.2xcytpi) **31**

[**The Logisim CPU**](#_heading=h.1ci93xb) **31**

[Write memory](#_heading=h.3bg54rfck7p9) 31

[Use immediate](#_heading=h.fdy6onp2m88i) 31

[Write ACC](#_heading=h.al41604a3hxn) 31

[Branch always](#_heading=h.qzdbchg10mf6) 31

[Branch NE zero](#_heading=h.e85v9q5e26k0) 31

[HALT](#_heading=h.mz59y6ky3c5) 31

[The Arithmetic logic unit](#_heading=h.2q81gaylck8v) 31

[**MIPS Assembly Language compared to the Logisim CPU**](#_heading=h.3whwml4) **32**

[Similarities and Differences](#_heading=h.53azm28e0d6l) 32

[**Links to programming in a high-level language**](#_heading=h.2bn6wsx) **33**

[**Conclusion**](#_heading=h.qsh70q) **34**

[**References**](#_heading=h.1pxezwc) **35**

[**Appendix A: The Logisim Code**](#_heading=h.147n2zr) **36**

[**Appendix B: The MIPS Code**](#_heading=h.3o7alnk) **55**

[**Appendix C: Completed Marking Grid**](#_heading=h.23ckvvd) **58**

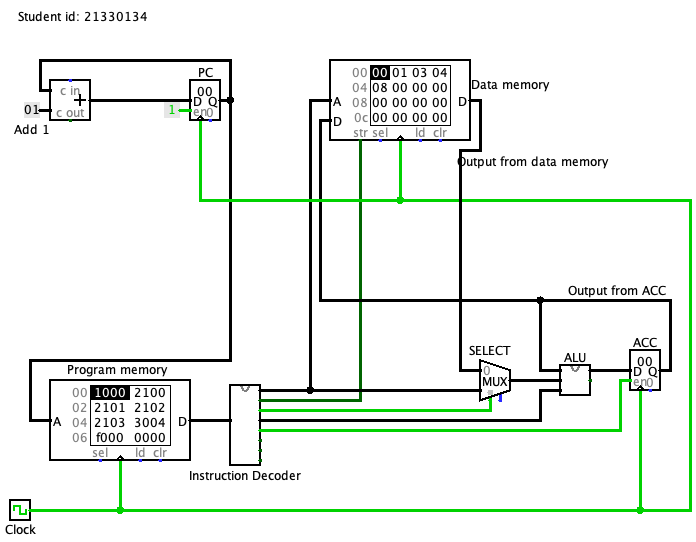
# Introduction

This report shows experimentation and completing various tasks in two different environments, the MARS and MIPS environment and Logisim environment, with the goal to compare them through experience with both. An objective in Logisim is to replicate how a cpu works, and in MIPS to be able to do the same commands as it. Then there will be comparison of both to see how they compare and also a comparison to another high level language. In doing this I will know the differences and similarities so if I experience any other low level languages I am more ready and will know the nuances of this type of programming.

# Logisim CPU Simulation

## Testing the CPU

### Data loaded



### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

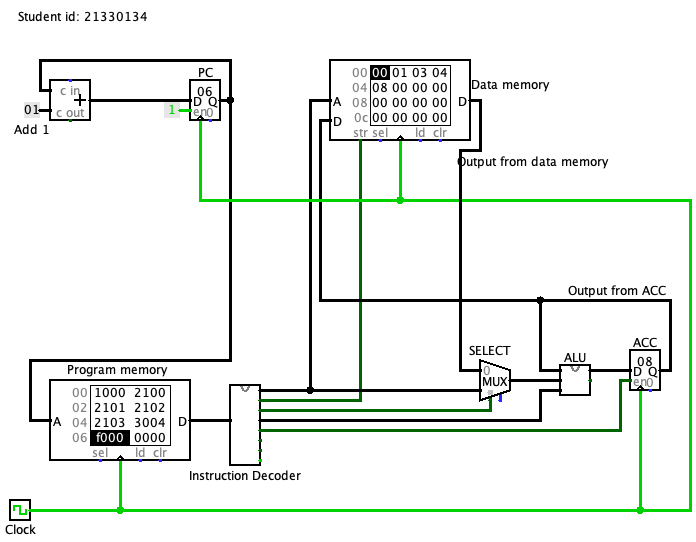
### 

### 

### 

### 

### End of Program

**

Total of digits in decimal is … 8

In hexadecimal is …8

This task is complete because the data memory at 04 shows 08 as expected.

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

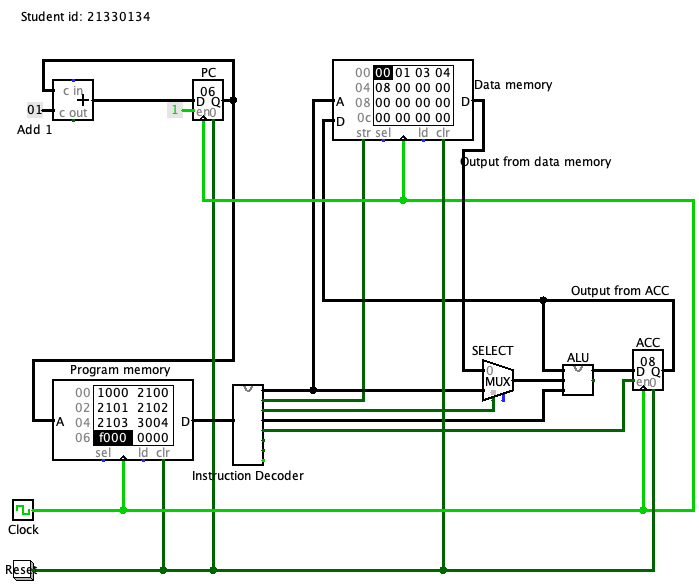
## 

## 

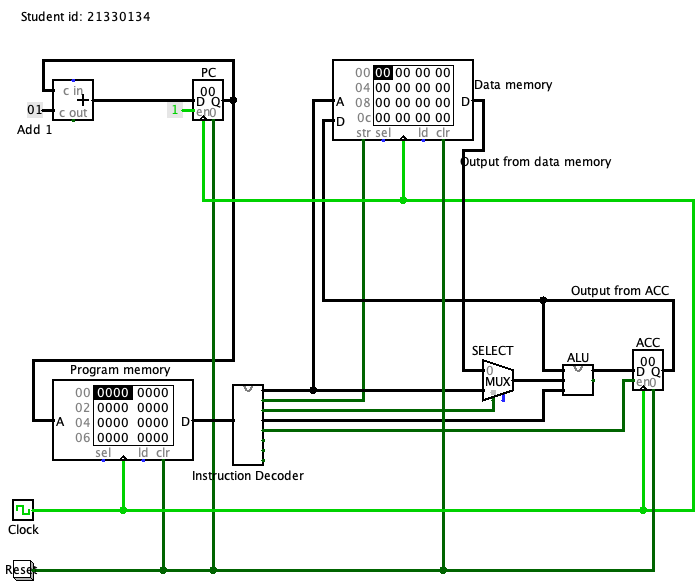
## Amendments to the CPU

### Implementing a reset button

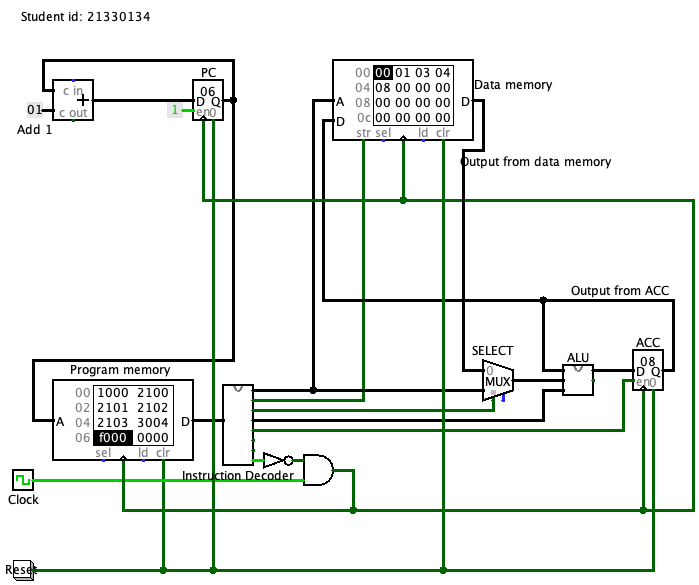
#### Data loaded



#### After reset

**

### Adding the halt instruction



The halt task is complete.

### Adding functionality to the ALU

#### Multiplication

These tests show 3 different situations to test the functionality of the multiplication and data handling.

TEST ONE:

Contains 0 in data

memory:data

00:00

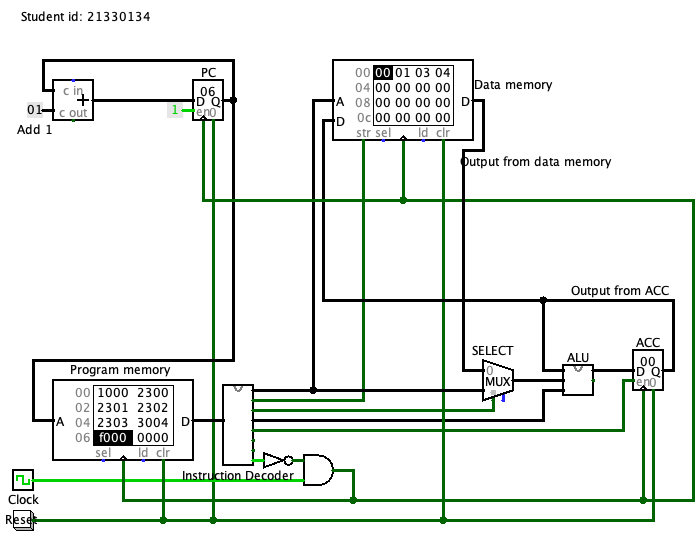
01:01

02:03

03:04

expected output in memory 04:0, decimal 0

actual output:



TEST TWO:

Does not contain 0 and is within the memory limit

memory:data

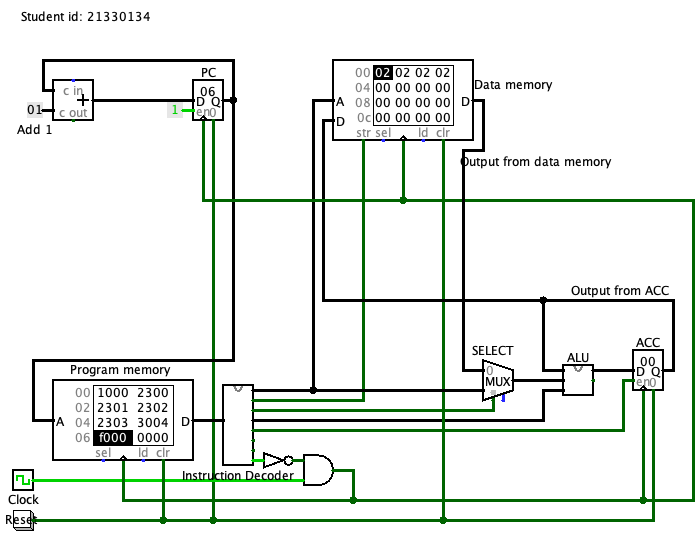
00:02

01:02

02:02

03:02

expected output in memory 04:10, decimal 16

actual output:

I ran into an issue here as what is happening is ((((0\*2)\*2)\*2)\*2) as the initial value in the ACC is 0, and is making my sum 0 what I need to do is ((((0+2)\*2)\*2)\*2) which will give me my expected result.

It did not affect my first test as I multiplying by 0 anyway at the start [ ((((0\*0)\*1)\*3)\*4) ] so it was not noticeable in the ACC

TEST ONE REDONE:

Contains 0 in data

memory:data

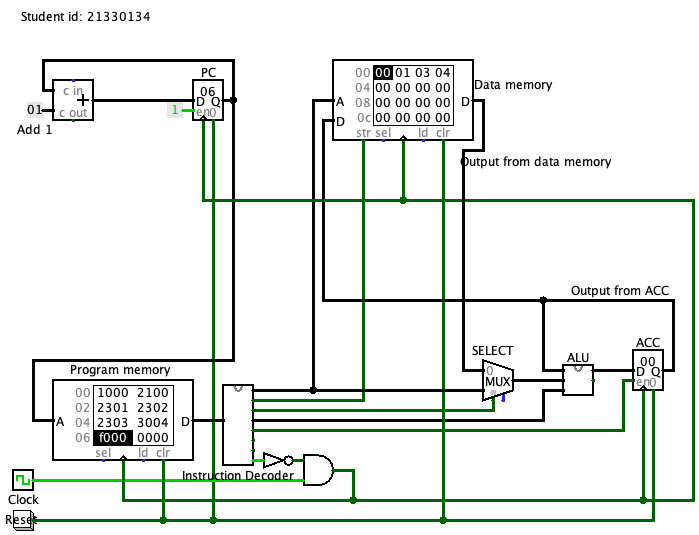
00:00

01:01

02:03

03:04

expected output in memory 04:0, decimal 0

actual output:

TEST TWO REDONE:

Does not contain 0 and is within the memory limit

memory:data

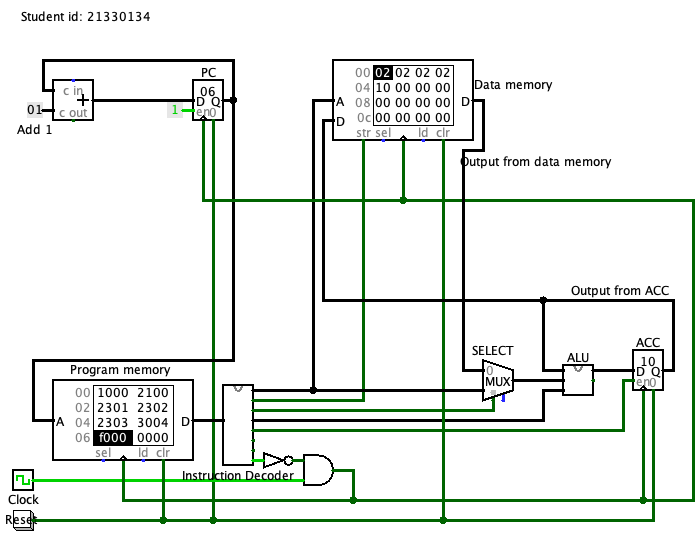
00:02

01:02

02:02

03:02

expected output in memory 04:10, decimal 16

actual output:

TEST THREE:

Does not contain 0 and is outside the memory limit

memory:data

00:01

01:01

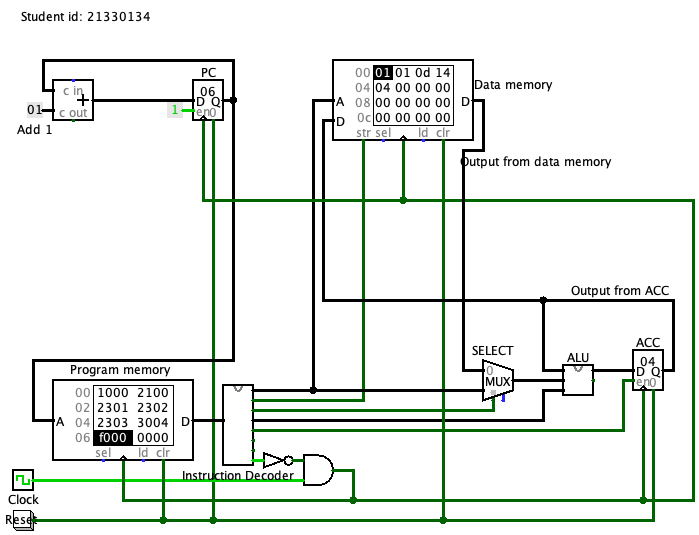
02:0d

03:14

expected output in memory 04:04, decimal 4

the output should be 104 and 260 but due to overflow it cuts off the 1 in the third position

actual output:

**

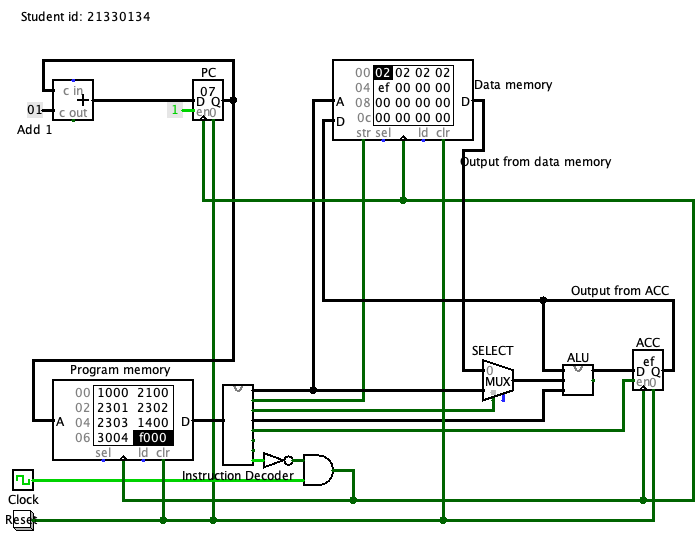
#### Inversion

TEST ONE:

Inverting

original before inversion hex 10 binary 0001 0000

expected output in memory 04:ef , decimal 223 binary 1110 1111

actual output:

#### Final ALU

### Adding functionality for branch instructions

#### Branch Always

The original test was creating a fibonacci sequence, because the only requirement was to show a loop that has been changed to make it simpler to understand in this report.

My programme is:

00:1000

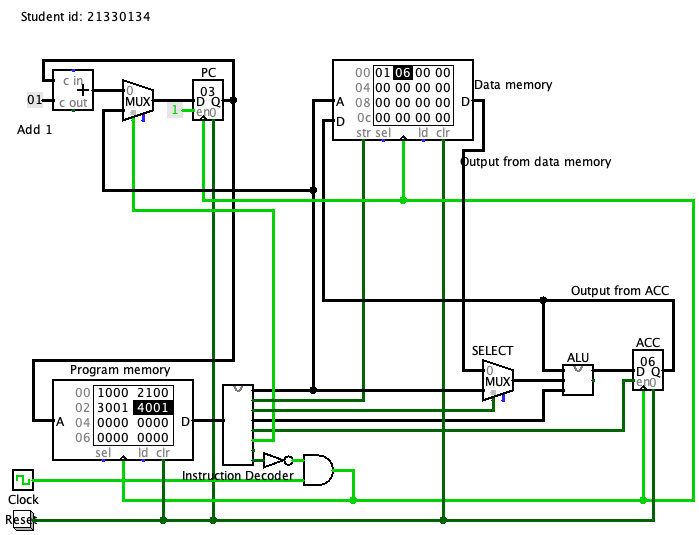
01:2100

02:3001

03:4001

This programme first sets the ACC to 0 then adds the value from data location 00

then it saves what is in the ACC to data location 01. Finally the branch always puts the PC to 01 and the programme repeats from the program memory location 01.



#### Branch Conditionally

Like previously, the original test was creating a fibonacci sequence, because the only requirement was to show a loop that has been changed to make it simpler to understand in this report.

My programme is:

00:1000

01:2001

02:2500

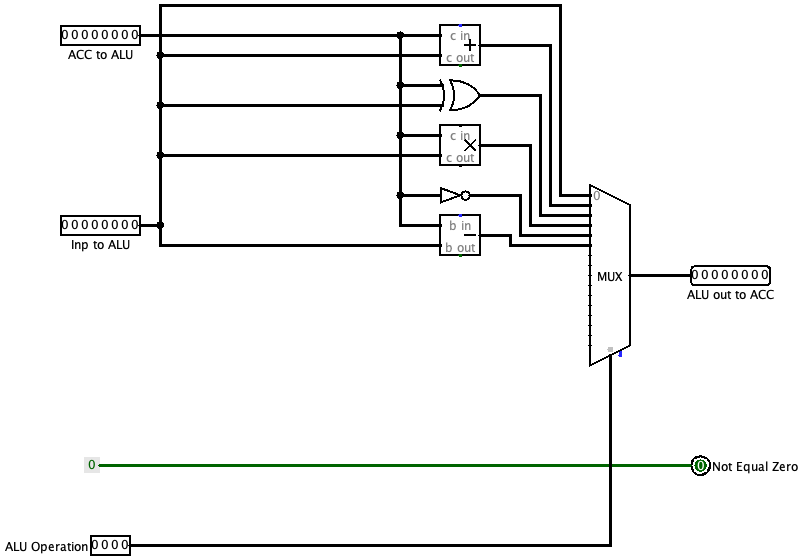
03:3001

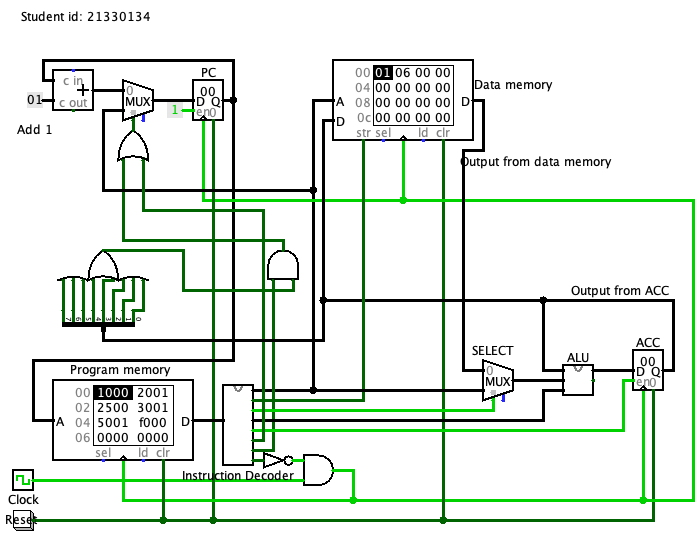
04:5001

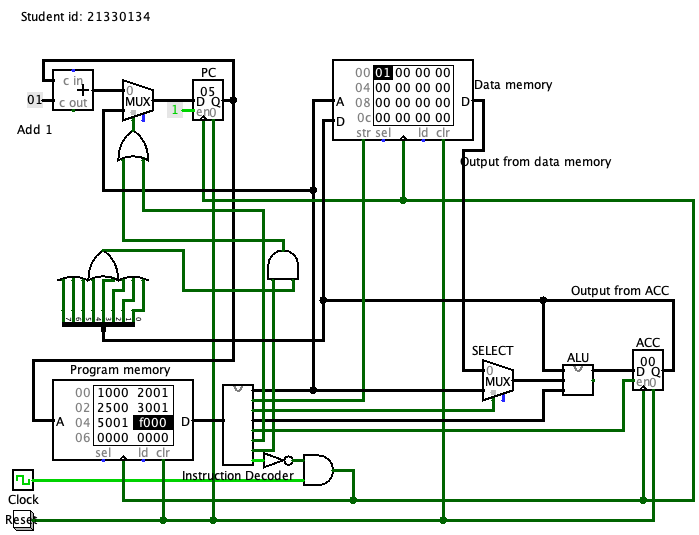
05:f000

This programme first sets the ACC to 0 then loads the value from data location 1

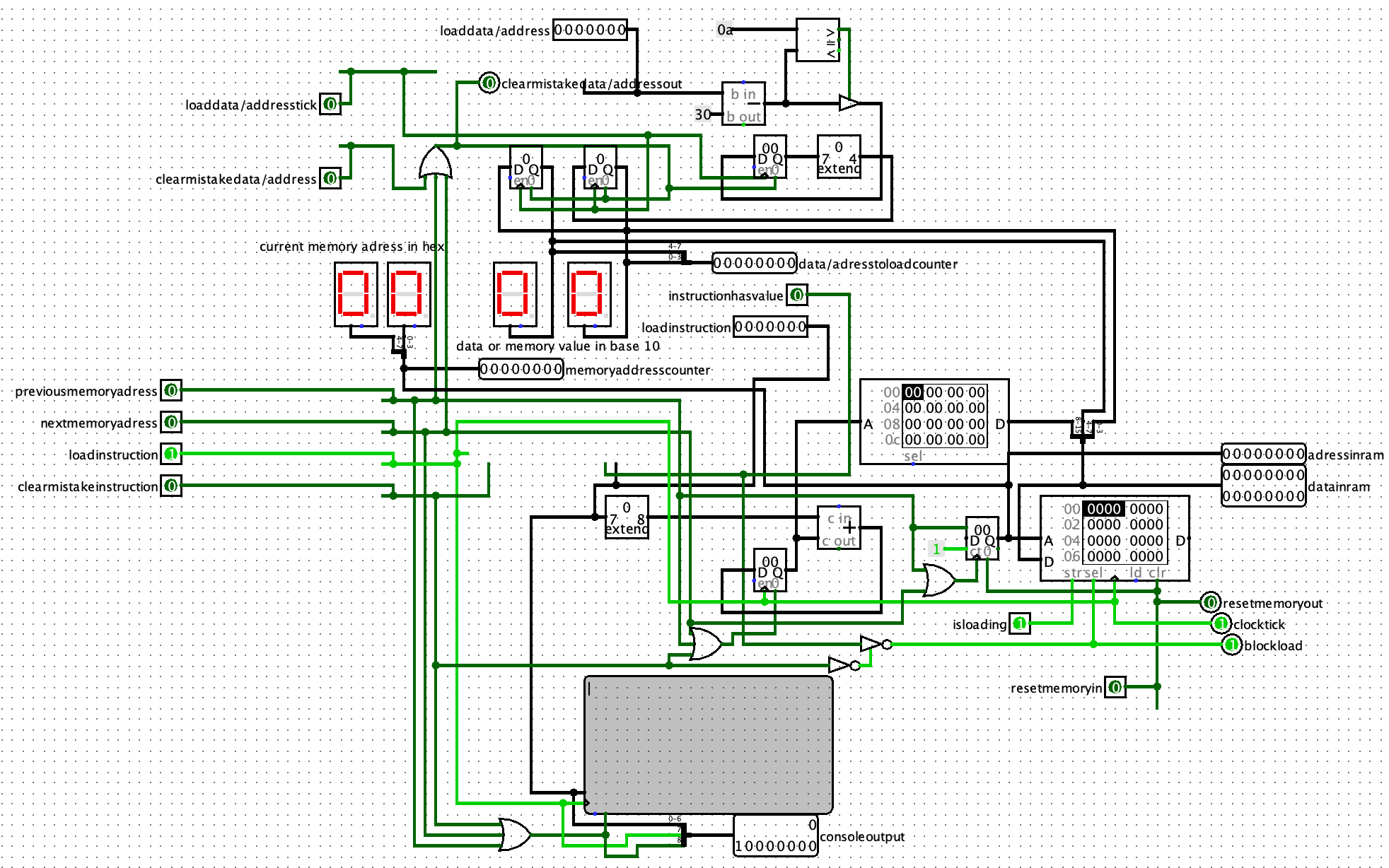
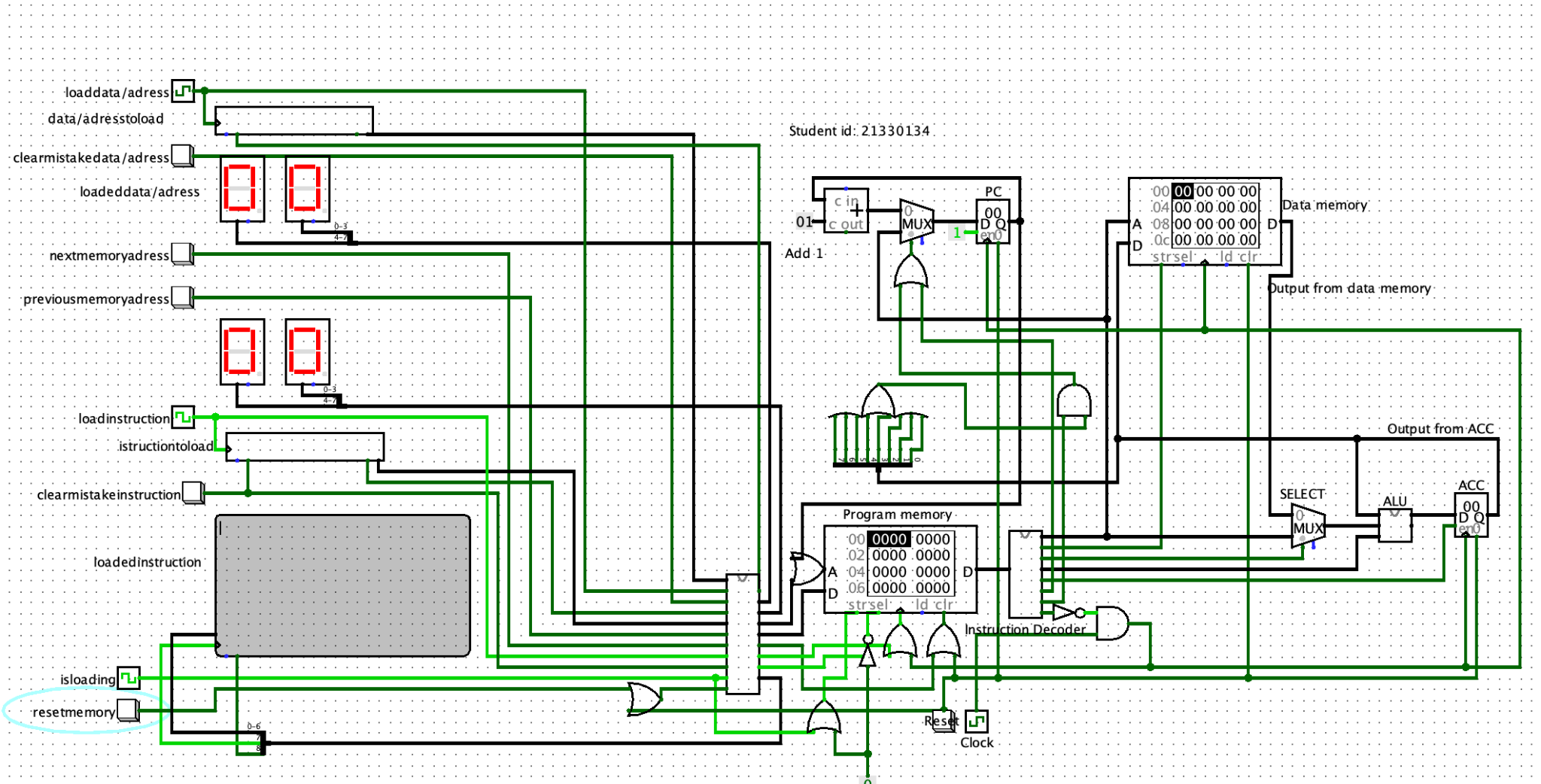
then it takes away what is in data location 0. Then it saves what is in the ACC to data location 01. Finally the branch always puts the PC to 01 and the programme repeats from the program memory location 01 unless it is 0 then the PC increments to 05 and the programme halts.







### Incorporating Input and Output devices



#### Making it easy

The way to make this easiest for the end user is to use buttons, words and make the input numbers denary. To do this I divided the input into two bits, the instructions and the data/memory location, using Logisim components for keyboard and console ([Logisim](http://www.cburch.com/logisim/index.html), Oct 2014). In the program there are codes to operate it. I have decoded theise above and given a brief explanation of what each does.

The number section takes in a raw number ascii value, reads it and decodes it into a number and saves it to the right display and any other values or numbers get removed. As ascii numbers from 1-9 in Logisim are 31,32,..,39 I subtract 30 and if it is less than 10 I keep it as a number. When a new number gets loaded in it pushes the number currently in the left display off and saves it in the right display, technically it is saved into a register not a display but it makes it easier for the user if they can see it. Also it does not load into the program memory until an instruction is loading as you can not have a value without an instruction.

The instruction section takes in certain words and decodes them, turning them into instructions that will be loaded into instruction memory along with the data or the address. It works by adding all the values of the characters together then looking up the value that is created in a ROM and if they match up it gives out the values needed for the command. or example, if stop is loaded and all the characters are added together it gives out the value c6 (hexadecimal) and in the roms memory at location c6 is f0 which is the command for halt and then this is loaded into the program memory with the 00 that will be in the number section (if it is not changed) additionally the characters are loaded into a console that will display the characters that have been loaded.

#### Breaking it down

##### Instruction layout

This is the layout of the data bits in the program memory with each nibble representing each hex character.

| 0 1 2 3 | 4 5 6 7 | 8 9 10 11| 12 13 14 15 |

|IDU operation|ALU operation| Data or memory address location|

##### Instruction Decoding Unit (IDU) instruction list

The Instruction Decoding Unit takes the first nibble and then does one of the following operations.

0 - does nothing

1 - write acc immediate, use immediate

2 - write ACC (using memory defined in Data or memory address location)

3 - write memory (using ACC defined in Data or memory address location)

4 - Branch always (to Data or memory address location)

5 - Branch != 0 (to Data or memory address location AND ACC != 0)

6 - Halt

##### ALU Instruction list

Input is the data that the user put into the program and output is the value that will be saved to the accumulator.

0 - input to output (in ALU)

1 - input + ACC =output

2 - input XOR ACC = output

3 - input \* ACC = output

4 - ACC NOT = output

5 - ACC - input = output

##### The Instructions (and the memory locations)

| Plain text instruction | Command in ACC | Place in memory |
| --- | --- | --- |
| stop | f0 | c6 |
| intial (consistent mistake) | 10 | 81 |
| loop | 40 | ba |
| loopz | 50 | 34 |
| load | 20 | a0 |
| add | 21 | 29 |
| take | 25 | a5 |
| multi | 23 | 2b |
| xor | 22 | 59 |
| not | 24 | 51 |

##### Buttons

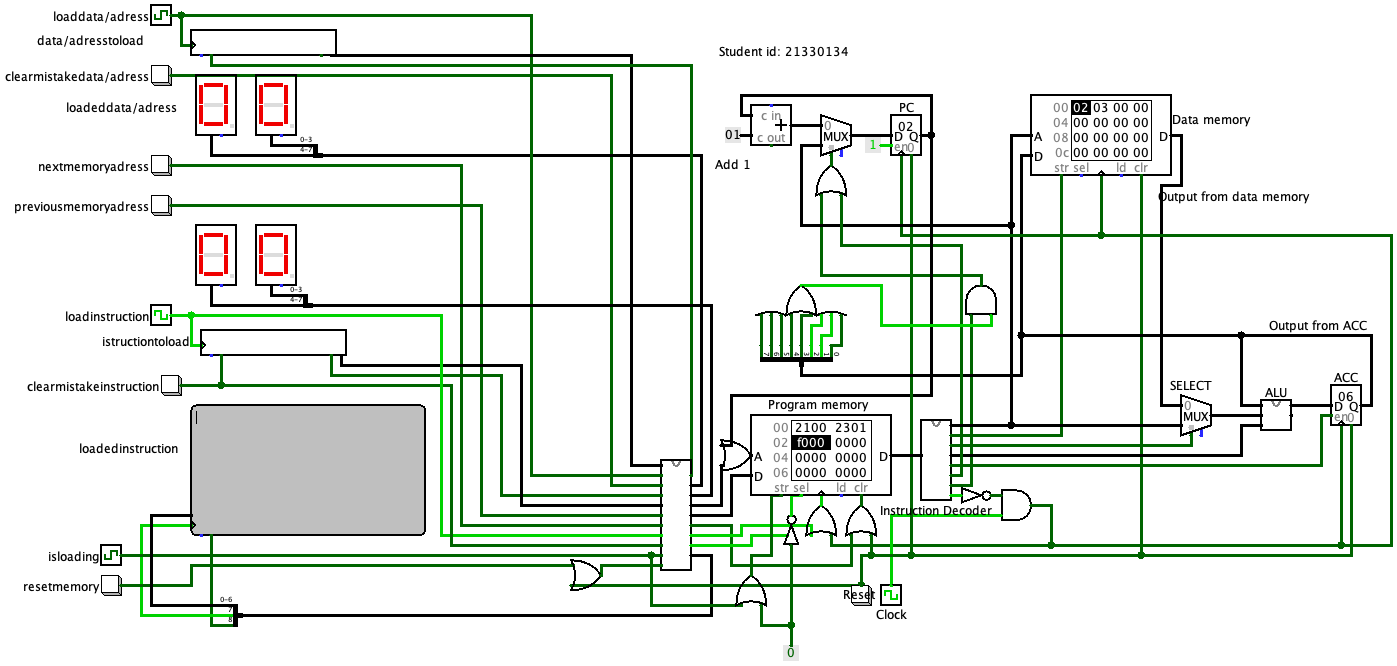
When loading the user has to choose the position in memory where the instruction using the buttons that move where to load the instructions. Additionally there is a display for the location of the memory point at where the instruction will be loaded. An issue at the moment is that the user will have to manually move the address back to 00 before trying to run the program although that is a relatively minor issue.

Both ascii fields have resets for the values they have put in so if the user makes an error they can reset easily. The program memory that they are writing to also has a reset switch so the user can start over again.

Additionally there is a clock that I have used as a switch to load switch whether or not the user is loading in a value, this will need to be turned off if they want to run the cpu simulation.

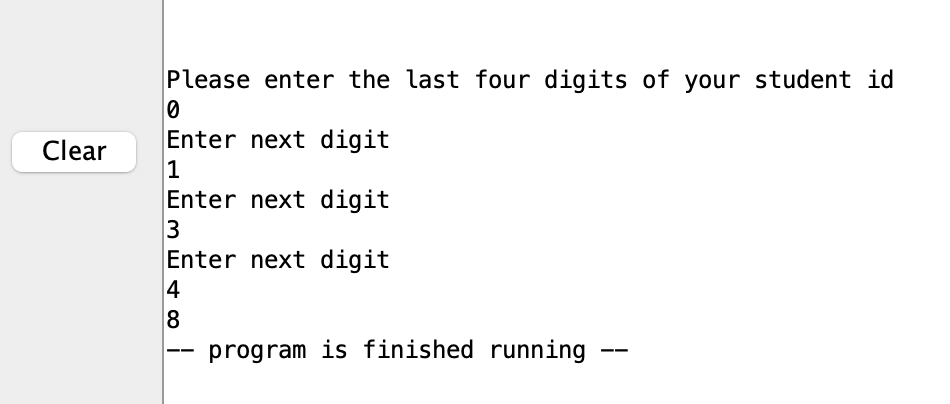
([Logisim](http://www.cburch.com/logisim/index.html), Oct 2014)

#### Loading a program and running it



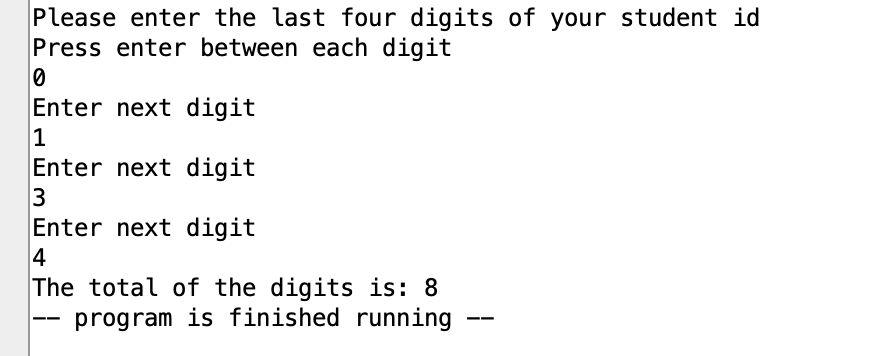
# MIPS Assembly Language Programming

## Testing the existing program

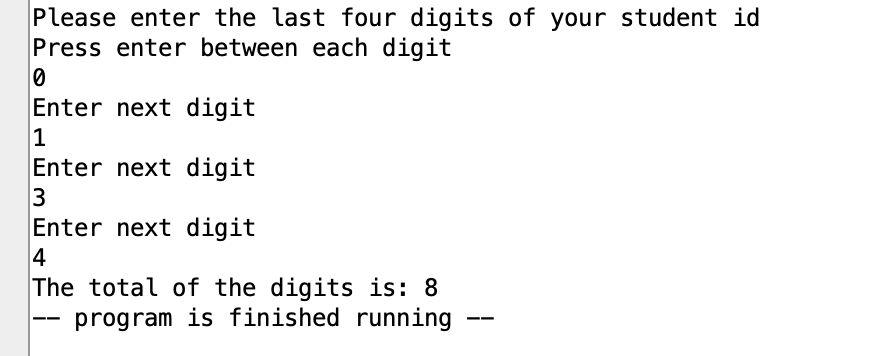
**

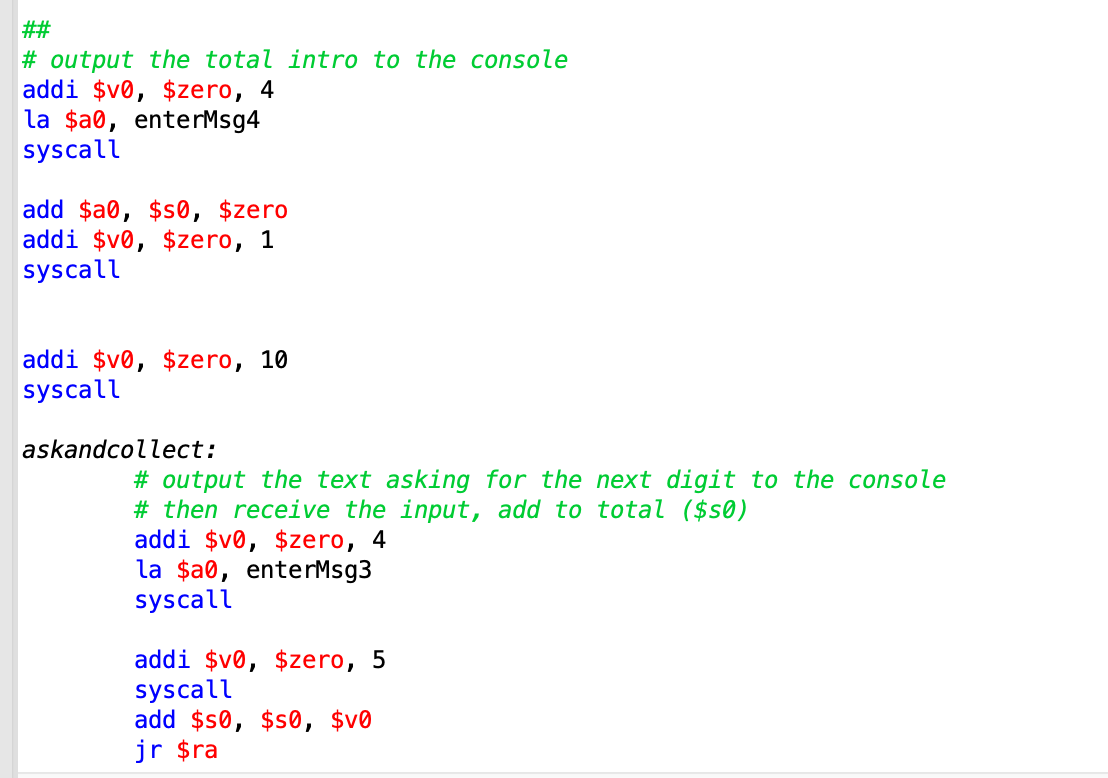
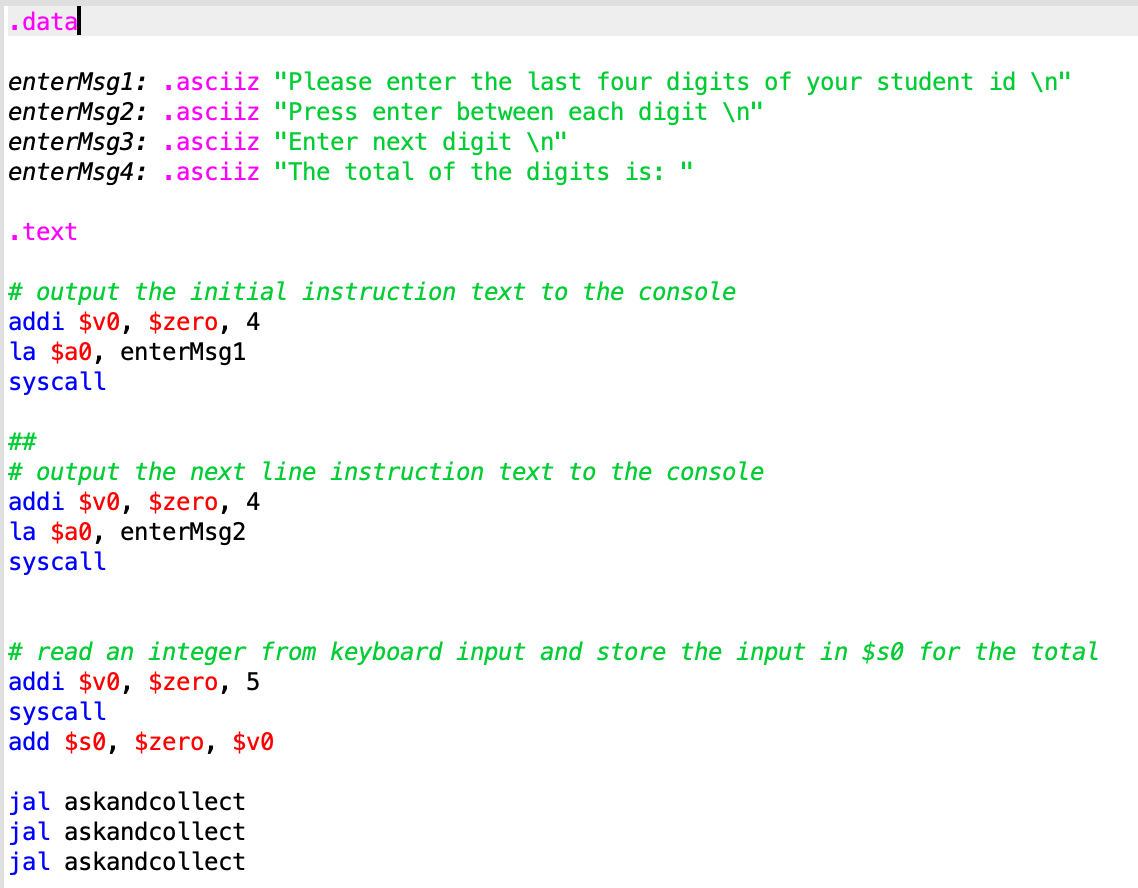
## Amendments to the program

### Improvements to messages output

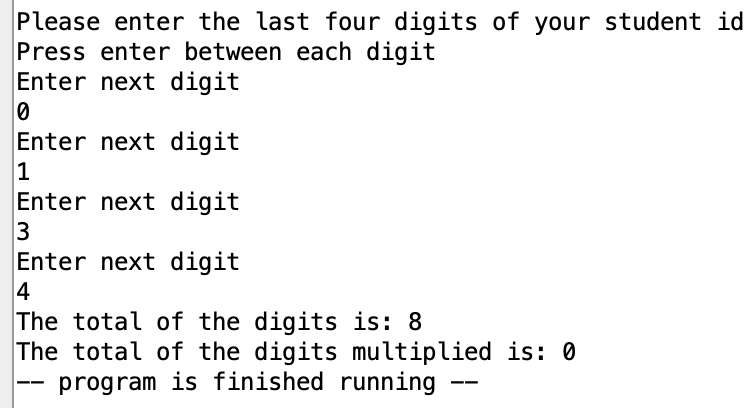
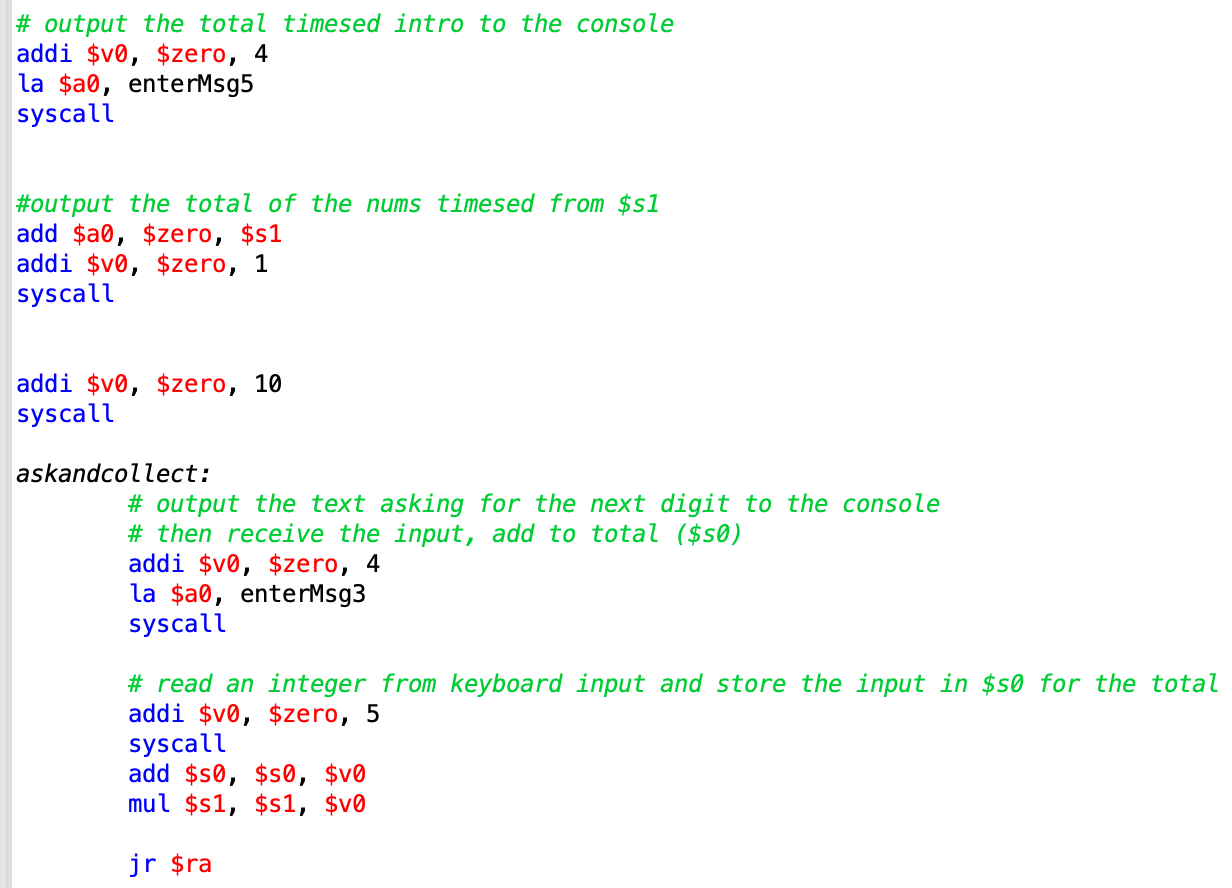
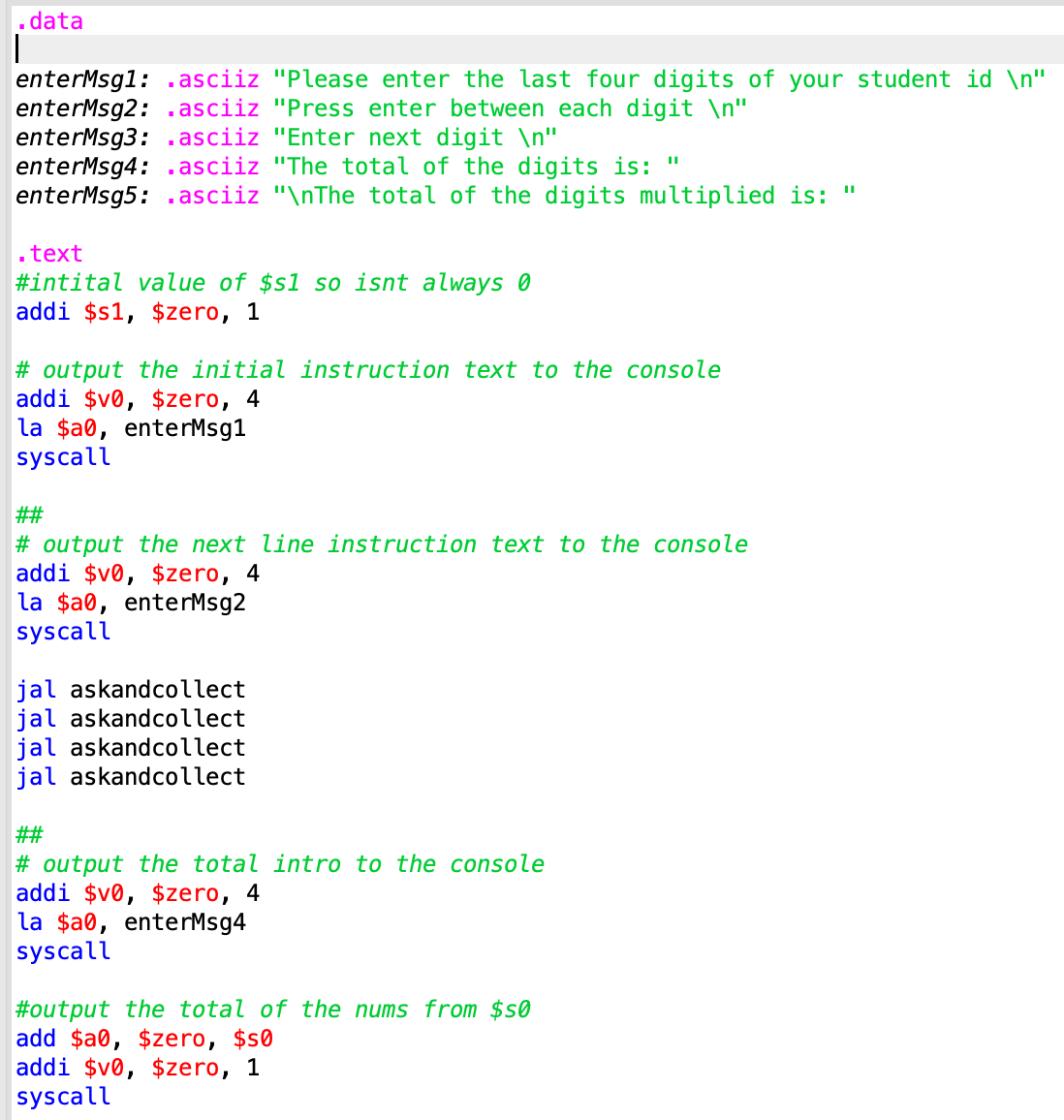


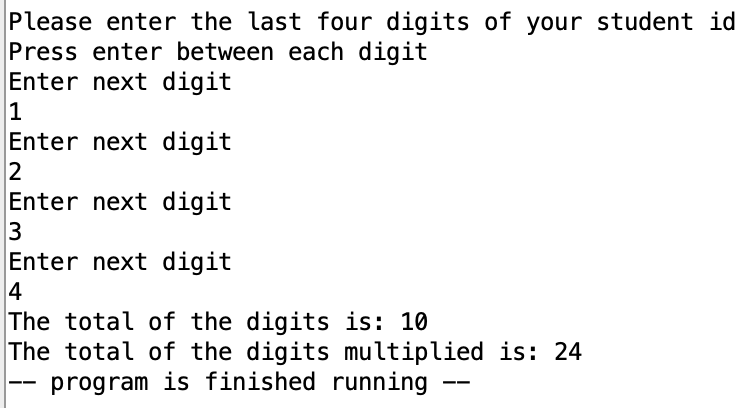
### Using a subroutine

**



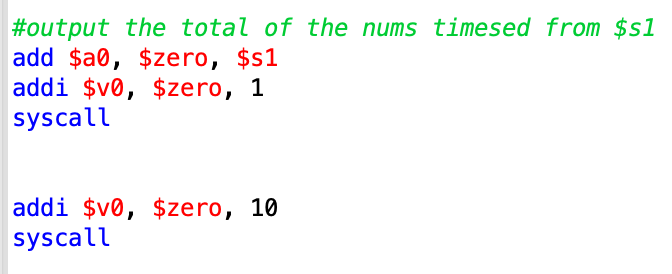
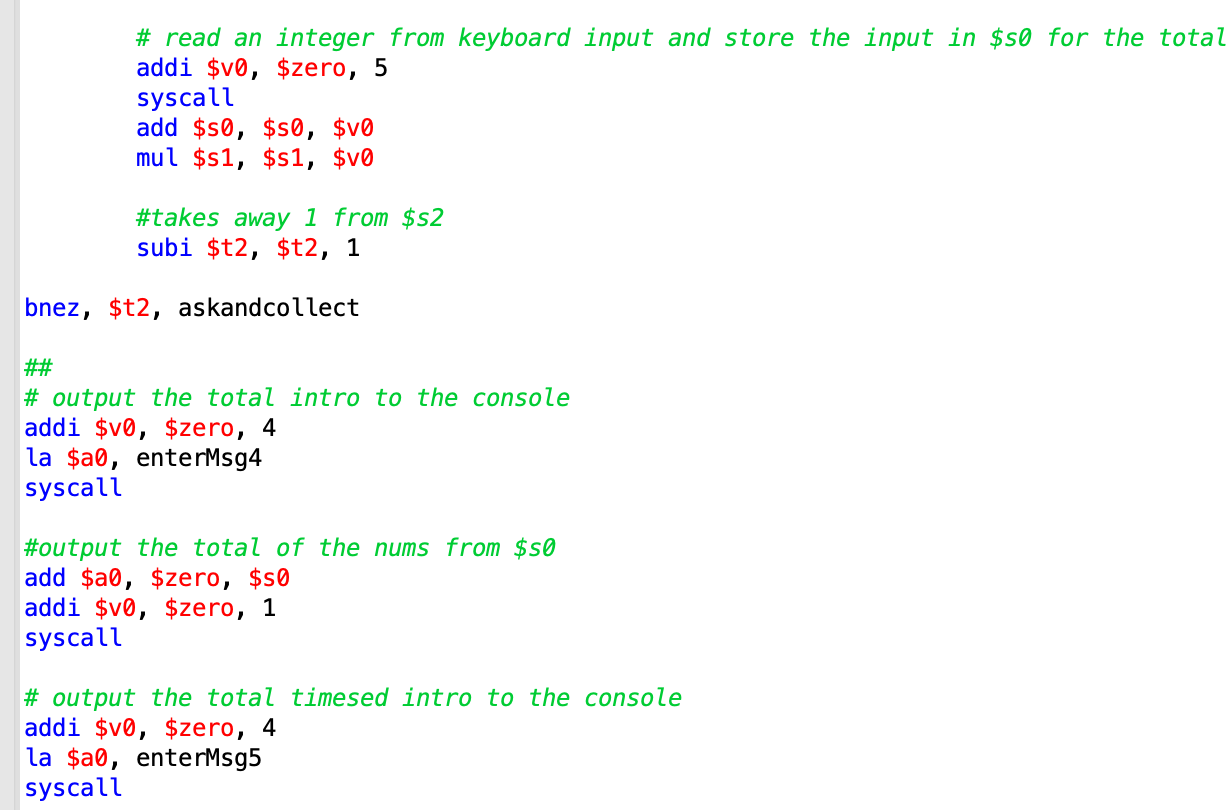
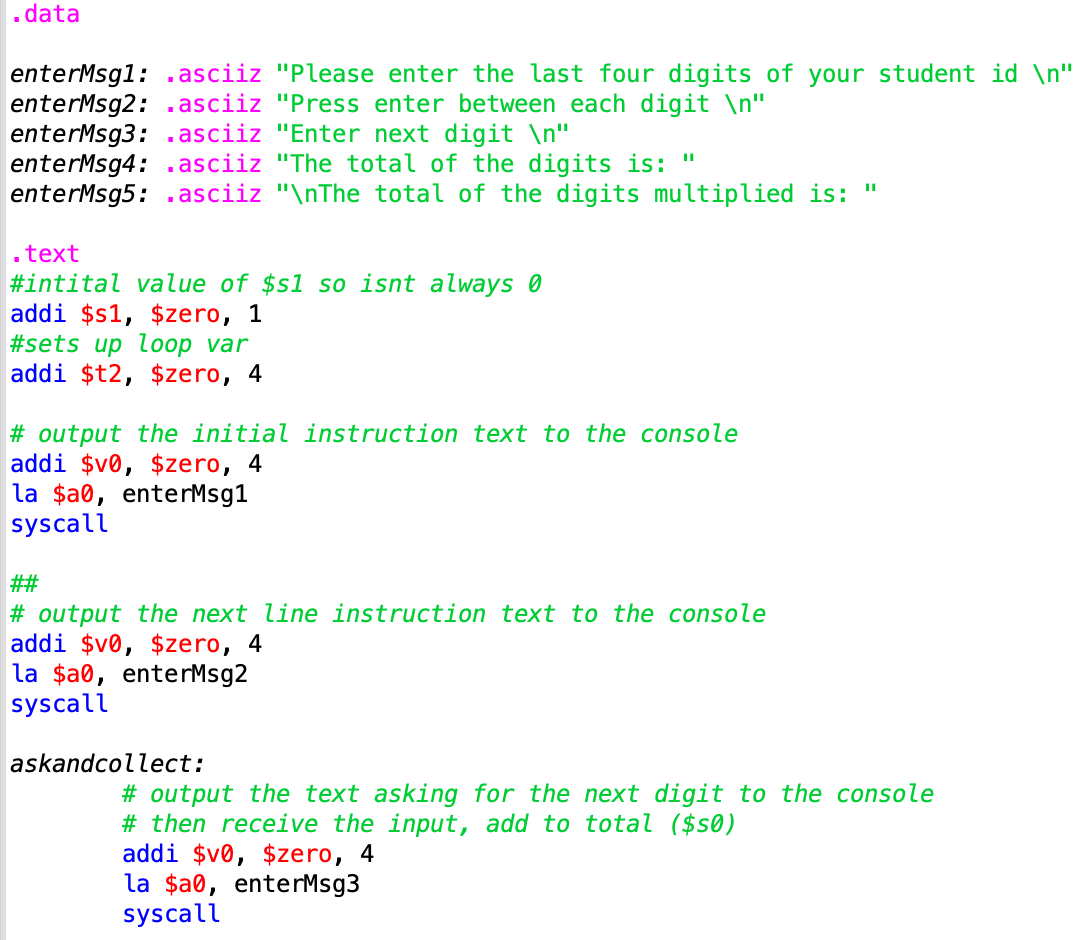
### Adding functionality

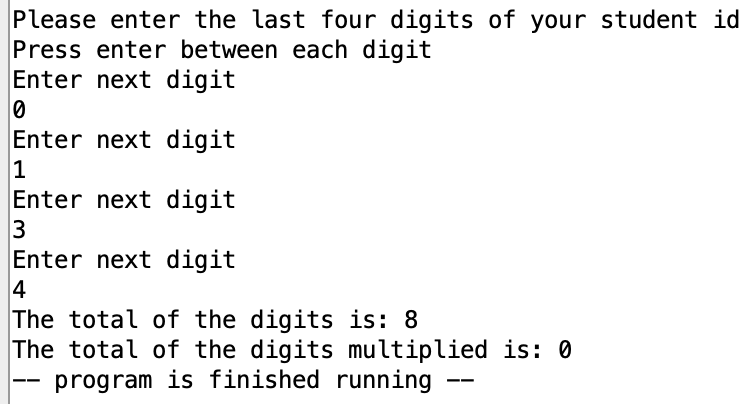
**



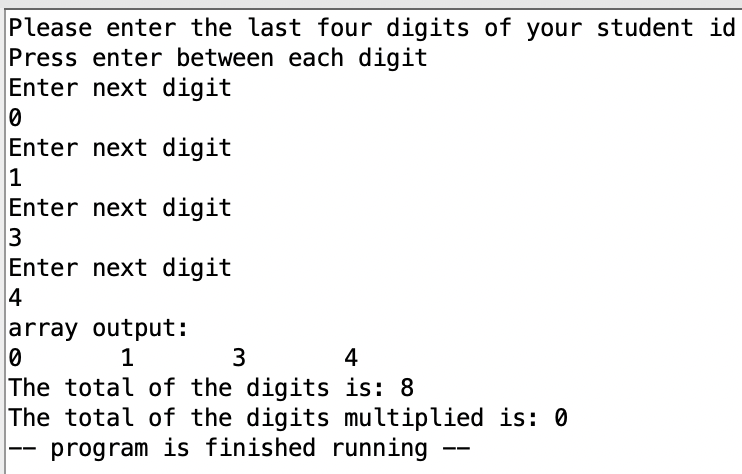
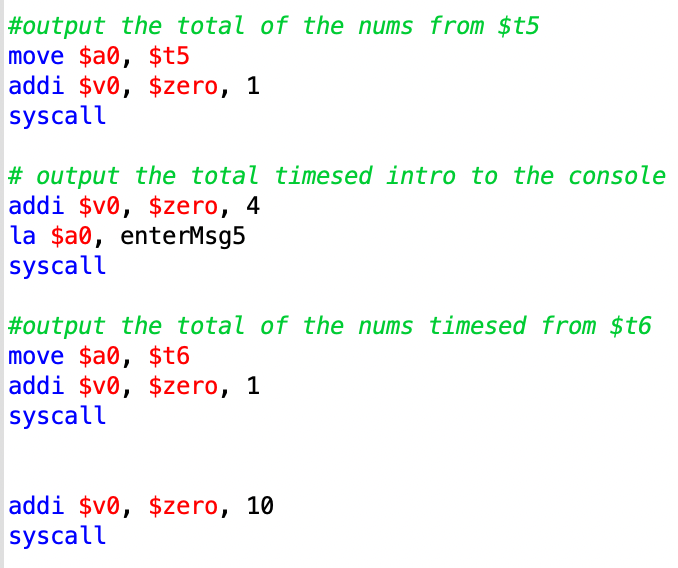
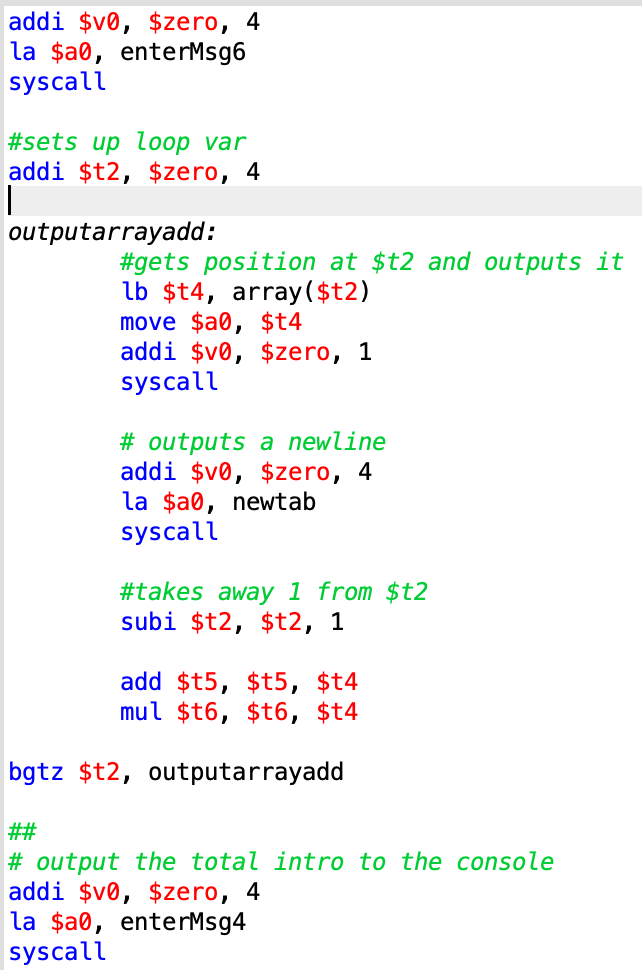
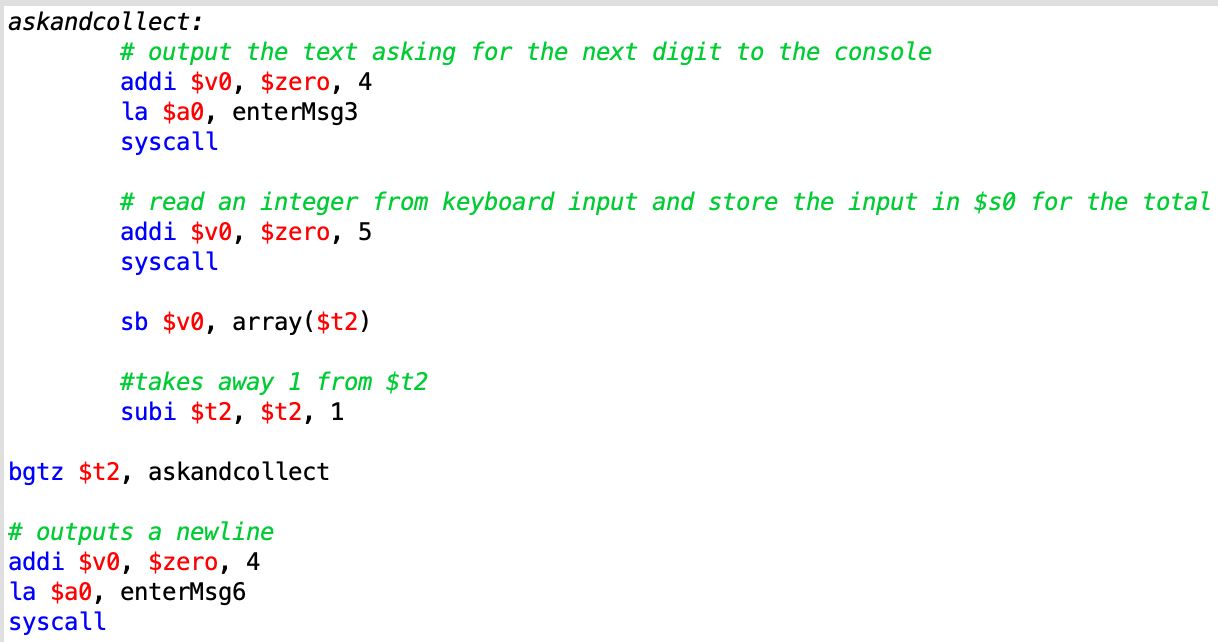
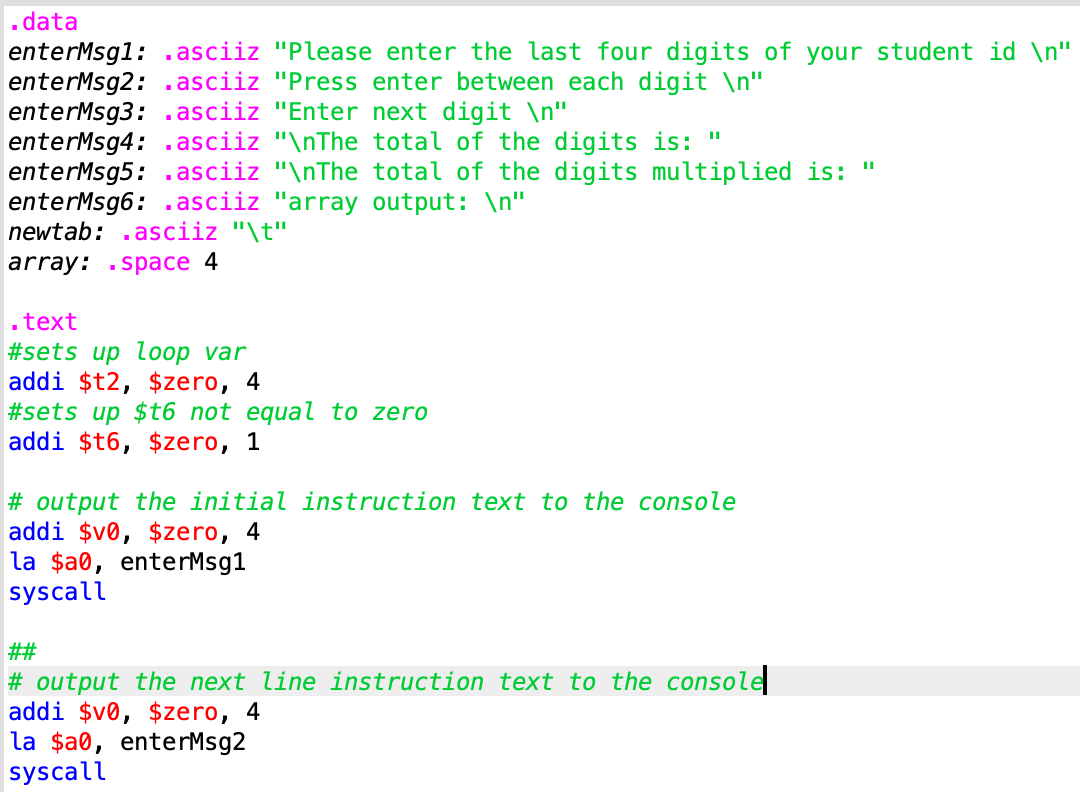
Because zero times anything is zero is zero the register $s1 needs to be 1 if a number is going to be multiplied to it inside the loop.

### Adding a loop





### Using an array



#### How arrays work

Arrays are lists of things that are of the same data type, this is useful if you need to iterate through data, take in a lot of data or have run out of register names for your program.

##### Initialization

To first initialize an array you first need to declare it in the data space at the top of the computer in format arrrayname: . space [number]

there is an array name then the .space reserves the amount of bytes specified by the number you have put.

##### Saving

To save into the array you use

lb $t0, arrayname($t1)

In this instruction lb means load byte and in this example $t0 is the data to be loaded into the array and $t1 is the position of where the data will be saved.

##### Reading

To save into the array you use

sb $t0, arrayname($t1)

In this instruction sb means save byte and in this example $t0 is the data to be read into the array and $t1 is the position of where the data will be read from.

(Amell Peralta, Dec 2014)

# Discussion

## The Logisim CPU

**The Instruction Decoder**

After loading the instructions into the program memory the clock ticks once the data enters the instruction decoder. The code that is passed into the instruction decoder is two bytes which is four nibbles, it is easier to talk about this program this way as each nibble represents a single hex character.

In the instruction decoder the first nibble gets decoded into a few sets of commands in the Instruction Decoder, the second nibble goes directly to the ALU and the last byte gets sent to other parts of the cpu. There is six commands;

#### Write memory

This sends the last byte that was in the instruction decoder to the data memory and saves it at the value that is in the program counter. The program counter keeps track of where the data memory is at and can be read to store an instruction at that point.

#### Use immediate

This sends the last byte that was in the instruction decoder to the accumulator to overwrite it. The accumulator which holds a value temporally that can be manipulated.

#### Write ACC

This overwrites the accumulator to the memory address defined in the last byte of the instruction, the memory address gets passed to the data memory then the data at that address gets loaded into the accumulator.

#### Branch always

Branches to the location defined in the last byte by changing the program counter to the value of the byte.

#### Branch NE zero

Branches to the location defined in the last byte by changing the program counter to the value of the byte, IF the accumulator is NOT equal to zero.

#### HALT

This ends the program.

### The Arithmetic logic unit

The second nibble that gets put into the program memory and then into the Arithmetic logic unit then, into a multiplexer to choose between six operations that will manipulate the two inputs one from the program memory and one from the accumulator. They are all quite simple like the add, minus and the multiply but the simple nature of the Arithmetic logic unit makes it easy to add more features. One of the more complicated ones is the XOR, it works by comparing each input one bit at a time and according to the table below it gives an output. The output is then saved into the accumulator.

| XOR TABLE | | |
| --- | --- | --- |
| program memory | accumulator | output |
| 1 | 1 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

## MIPS Assembly Language compared to the Logisim CPU

To compare both environments I have written a program in each that does the same thing, I am aware that the Logisim uses memory and MIPS uses registers but the way that I have used them it does not make a difference as they are both just saving numbers. If letters or words tried to be used it would not work for Logisim as the current program does not support it.

To compare the programs to each other I am going to first load a number into a memory position, then load a value into a temp location, add them together then overwrite the first memory location with the result and end the program.

| column number | MIPs | Logisim(in program memory) | What they do |
| --- | --- | --- | --- |
| 1 | .text | - | This tells where the compiler the start of the program is |
| 2 | addi $t0, $zero, 4 | 1004 | Sets the value of a memory position to 4 |
| 3000 |
| 3 | addi $t0, $t0, 4 | 1004 | Adds 4 to the saved memory position then overwrites it with the new value |
| 2100 |
| 3000 |
| 4 | addi $v0, $zero, 10 | f000 | Ends the program |
| syscall |

### 

### 

### 

### 

### 

### Similarities and Differences

1.

Logisim does not need to define where the instructions start as there are only instructions in program memory whereas MIPs can have other data in its program and it needs to differentiate between them so it does not try to read text as an instruction.

2

.MIPs can set a value to a memory position in one line but Logisim needs 2.

3.

Logisim needs to load a value into a temporary position (the ACC), then add it to a value in memory then save it all separately, MIPs does not, it just saves directly to the registers.

4.

MIPs needs to load commands into specific memory locations then execute them where Logisim only needs one line, this is because sometimes when syscall is run it takes in multiple values from a few different memory locations.

When making programs in Mips and you make a mistake it does not matter as much as it tells you exactly what line there is an issue on and there is comments to explain what you have done so when you come back to it you know exactly what you were doing and it is easier to swap in and out code.

Logisim if it messes up its just trial and error until you find out what is wrong so in longer programs it will be much easier to make a mistake additionally the commands and data AND the memory locations are all numbers very close to each with no comments. This makes it harder to read and easier to mix them up and impossible to actually program if you do not have codes for the instructions in front of you or are very familiar with them.

## Links to programming in a high-level language

Security is what I have chosen to focus on as in the modern world this is important as if anything is programmed on unstable roots it will infect anything you are trying to make, to make this happen the correct tools and environments must be picked, also where to host their downloads too. The two I am comparing the MARS environment that is being used in this report to a common java extension Log4J.

One issue I had with the MARS environment intillaily is that the website that we were told to get from ([MARS MIPS simulator - Missouri State University](http://courses.missouristate.edu/kenvollmar/mars/)) is that the download is from August 2014. It has been nearly 8 years since it had been released so I was worried about compatibility with my mac as it is one of the new M1 chips and runs on a different architecture to X86 and also apple forcing incompatibility but has been fine for the time being. Although in the future as software changes it may be incompatible in the future and from the web page it does not look like anyone is still updating it and it is just something someone is running because they forgot about it.

Additionally there are massive security risks with this website (Missouri State University, 2017) if the Computer Architecture course every year sends students to this page to download this. it is just a HTTP site (PowerCert Animated Videos, Dec 2018) and someone in a class or the library after the seminar could perform a man-in-the-middle attack (Computerphile, Oct 2015) and get them to download a replica of the program, also the programs that the students create and share could also have malware in them too, further spreading the issue.

This is an issue with using low level code and editors as in real world they are not used very often and when they are used there are better tools but more complicated and are pay to use as they are so niche.

However java has a massive community and developers so the likelihood of this happening is much less and software support and versions are much more up to date, but it is not a silver bullet as it is such a big community some things are overlooked and when the issue is finally spotted it is a massive issue. For example Log4J which is a logging software that records all the inputs that the user does and saves it into a file for the developer to look at. It is very useful for looking at attacks on programs and diagnosing issues, recently it was discovered you could manipulate log4J and the other systems supporting it to run and load code without even talking to the program that you were initially using (Computerphile, Dec 2021). The reason why Log4J is comparable to the MARS environment is that its use is massively widespread and also is sometimes included preinstalled so it is almost a part of java already.

Whatever software you use can have issues but the ones referring to the Log4J exploit are manageable as there has been lots of information been put out about it and the module for it can even be completely removed, but if there was a exploit with the MARS environment we probably will not know for a couple years as the programming effort is not behind it as it is just not as usable as the other main languages.

To improve the security of MARS the download should absolutely be hosted on a secure webpage by the University and also checked and scanned before an attack happens.

# Conclusion

Logisim was enjoyable and also very understandable to me and I even implemented an input system that encodes the users input to actual usable instructions, although it is difficult to be neat with Logisim with more complicated programs and when mistakes happen. This is because you have to use your mouse to make lines and moving multiple lines, or a component that is connected to a few is difficult as they just go where they like and connect to whatever. However, actually seeing lines light up and values change makes it easy to rapidly understand and change the program, making it much better for new people to programming, just frustrating if you want to do a bit more complicated stuff.

In my opinion, the MIPS is incredibly easy to make simple assembly, low level programs due to the well laid out code. The only major issue is that I do not trust the MARS environment, it is an old out of date program whose development stopped 6 years ago and is downloaded from a HTTP site which is vulnerable to attacks. All it needs is someone to run a scan on it and move the download to a HTTPS site and it would be fine but in the current state it is unacceptable. The language is good for me but for people newer to programming I would say they would struggle as it does deal a lot with memory locations.

The comparison was fine but they are not really the same thing until you have built up the Logisim program quite a bit and it still can not deal with letters and strings at all at the moment, whereas MIPS just has that ready to go.

In my opinion Logisim will be good for familiarizing people with how a CPU and memory works and even simple board design, but if your goal is to teach or learn assembly the main way should be MIPS.

# References

Amell Peralta, (Dec 2014) YouTube at URL: [MIPS Tutorial 27 Arrays](https://www.youtube.com/watch?v=BfHcogmKM20)

(Accessed 12 Jan 2021)

Amell Peralta, (Dec 2014) YouTube at URL: [MIPS Tutorial 28 Printing an Array with a While Loop](https://www.youtube.com/watch?v=Vb8kuvxc4NE)

(Accessed 12 Jan 2021)

Computerphile, (Oct 2015) YouTube at URL: [Man in the Middle Attacks & Superfish - Computerphile](https://www.youtube.com/watch?v=-enHfpHMBo4)

(Accessed 12 Jan 2021)

Computerphile, (Dec 2021) YouTube at URL: [Log4J & JNDI Exploit: Why So Bad? - Computerphile](https://www.youtube.com/watch?v=Opqgwn8TdlM)

(Accessed 12 Jan 2021)

Logisim (Oct 2014) Logisim Available at URL: [Logisim](http://www.cburch.com/logisim/index.html)

(Accessed 12 Jan 2021)

Missouri State University, (2017) MARS (MIPS Assembler and Runtime Simulator) at URL: [MARS MIPS simulator - Missouri State University](http://courses.missouristate.edu/kenvollmar/mars/)

(Accessed 12 Jan 2021)

PowerCert Animated Videos, (Dec 2018) YouTube at URL: [SSL, TLS, HTTP, HTTPS Explained](https://www.youtube.com/watch?v=hExRDVZHhig)

(Accessed 12 Jan 2021

# Appendix A: The Logisim Code

| <?xml version="1.0" encoding="UTF-8" standalone="no"?>  <project source="2.7.1" version="1.0">  This file is intended to be loaded by Logisim (http://www.cburch.com/logisim/).  <lib desc="#Wiring" name="0">  <tool name="Splitter">  <a name="facing" val="west"/>  <a name="appear" val="center"/>  </tool>  <tool name="Pin">  <a name="facing" val="north"/>  <a name="width" val="24"/>  <a name="tristate" val="false"/>  </tool>  <tool name="Probe">  <a name="radix" val="10signed"/>  </tool>  <tool name="Tunnel">  <a name="facing" val="east"/>  <a name="width" val="24"/>  <a name="label" val="RES"/>  </tool>  <tool name="Pull Resistor">  <a name="facing" val="east"/>  </tool>  <tool name="Clock">  <a name="facing" val="north"/>  </tool>  <tool name="Constant">  <a name="width" val="24"/>  <a name="value" val="0x0"/>  </tool>  </lib>  <lib desc="#Gates" name="1"/>  <lib desc="#Plexers" name="2"/>  <lib desc="#Arithmetic" name="3">  <tool name="Comparator">  <a name="mode" val="unsigned"/>  </tool>  </lib>  <lib desc="#Memory" name="4">  <tool name="ROM">  <a name="contents">addr/data: 8 8  0  </a>  </tool>  </lib>  <lib desc="#I/O" name="5"/>  <lib desc="#Base" name="6">  <tool name="Text Tool">  <a name="text" val=""/>  <a name="font" val="SansSerif plain 12"/>  <a name="halign" val="center"/>  <a name="valign" val="base"/>  </tool>  <tool name="Text">  <a name="text" val="Select"/>  </tool>  </lib>  <main name="CPU main"/>  <options>  <a name="gateUndefined" val="ignore"/>  <a name="simlimit" val="1000"/>  <a name="simrand" val="0"/>  </options>  <mappings>  <tool lib="6" map="Button2" name="Menu Tool"/>  <tool lib="6" map="Button3" name="Menu Tool"/>  <tool lib="6" map="Ctrl Button1" name="Menu Tool"/>  </mappings>  <toolbar>  <tool lib="6" name="Poke Tool"/>  <tool lib="6" name="Edit Tool"/>  <tool lib="6" name="Text Tool">  <a name="text" val=""/>  <a name="font" val="SansSerif plain 12"/>  <a name="halign" val="center"/>  <a name="valign" val="base"/>  </tool>  <sep/>  <tool lib="0" name="Pin">  <a name="tristate" val="false"/>  </tool>  <tool lib="0" name="Pin">  <a name="facing" val="west"/>  <a name="output" val="true"/>  <a name="labelloc" val="east"/>  </tool>  <tool lib="1" name="NOT Gate"/>  <tool lib="1" name="AND Gate"/>  <tool lib="1" name="OR Gate"/>  </toolbar>  <circuit name="CPU main">  <a name="circuit" val="CPU main"/>  <a name="clabel" val=""/>  <a name="clabelup" val="east"/>  <a name="clabelfont" val="SansSerif plain 12"/>  <wire from="(970,580)" to="(1340,580)"/>  <wire from="(860,360)" to="(980,360)"/>  <wire from="(500,630)" to="(680,630)"/>  <wire from="(1310,520)" to="(1350,520)"/>  <wire from="(160,640)" to="(210,640)"/>  <wire from="(1260,540)" to="(1260,570)"/>  <wire from="(350,490)" to="(520,490)"/>  <wire from="(710,640)" to="(750,640)"/>  <wire from="(890,650)" to="(890,670)"/>  <wire from="(840,310)" to="(840,390)"/>  <wire from="(970,600)" to="(990,600)"/>  <wire from="(750,590)" to="(750,630)"/>  <wire from="(740,580)" to="(740,620)"/>  <wire from="(750,510)" to="(950,510)"/>  <wire from="(200,160)" to="(210,160)"/>  <wire from="(970,610)" to="(980,610)"/>  <wire from="(670,670)" to="(670,720)"/>  <wire from="(750,510)" to="(750,560)"/>  <wire from="(810,610)" to="(820,610)"/>  <wire from="(1160,670)" to="(1370,670)"/>  <wire from="(480,640)" to="(680,640)"/>  <wire from="(830,640)" to="(830,650)"/>  <wire from="(630,600)" to="(680,600)"/>  <wire from="(880,610)" to="(880,620)"/>  <wire from="(760,210)" to="(950,210)"/>  <wire from="(780,430)" to="(780,460)"/>  <wire from="(860,430)" to="(860,460)"/>  <wire from="(1080,290)" to="(1080,550)"/>  <wire from="(160,640)" to="(160,730)"/>  <wire from="(240,500)" to="(470,500)"/>  <wire from="(170,630)" to="(210,630)"/>  <wire from="(200,500)" to="(240,500)"/>  <wire from="(820,260)" to="(820,340)"/>  <wire from="(910,630)" to="(1020,630)"/>  <wire from="(190,280)" to="(480,280)"/>  <wire from="(820,260)" to="(840,260)"/>  <wire from="(460,650)" to="(680,650)"/>  <wire from="(750,640)" to="(750,680)"/>  <wire from="(1040,450)" to="(1040,490)"/>  <wire from="(470,500)" to="(470,660)"/>  <wire from="(750,590)" to="(770,590)"/>  <wire from="(710,620)" to="(740,620)"/>  <wire from="(140,670)" to="(670,670)"/>  <wire from="(1260,450)" to="(1390,450)"/>  <wire from="(1120,290)" to="(1120,350)"/>  <wire from="(810,240)" to="(840,240)"/>  <wire from="(480,280)" to="(480,640)"/>  <wire from="(190,430)" to="(210,430)"/>  <wire from="(210,460)" to="(220,460)"/>  <wire from="(140,700)" to="(210,700)"/>  <wire from="(840,450)" to="(850,450)"/>  <wire from="(820,430)" to="(830,430)"/>  <wire from="(830,440)" to="(840,440)"/>  <wire from="(850,460)" to="(860,460)"/>  <wire from="(1040,270)" to="(1040,450)"/>  <wire from="(1180,300)" to="(1200,300)"/>  <wire from="(1260,540)" to="(1280,540)"/>  <wire from="(710,650)" to="(760,650)"/>  <wire from="(1030,340)" to="(1030,540)"/>  <wire from="(230,170)" to="(230,180)"/>  <wire from="(810,430)" to="(810,460)"/>  <wire from="(630,190)" to="(630,600)"/>  <wire from="(870,650)" to="(870,680)"/>  <wire from="(230,250)" to="(230,270)"/>  <wire from="(780,720)" to="(780,740)"/>  <wire from="(1360,660)" to="(1410,660)"/>  <wire from="(1230,530)" to="(1280,530)"/>  <wire from="(1410,350)" to="(1410,660)"/>  <wire from="(990,430)" to="(990,600)"/>  <wire from="(520,610)" to="(680,610)"/>  <wire from="(770,650)" to="(800,650)"/>  <wire from="(1160,290)" to="(1160,670)"/>  <wire from="(780,740)" to="(810,740)"/>  <wire from="(520,490)" to="(520,610)"/>  <wire from="(500,320)" to="(500,630)"/>  <wire from="(550,590)" to="(680,590)"/>  <wire from="(730,670)" to="(730,730)"/>  <wire from="(790,610)" to="(800,610)"/>  <wire from="(1050,620)" to="(1070,620)"/>  <wire from="(740,560)" to="(750,560)"/>  <wire from="(840,430)" to="(840,440)"/>  <wire from="(210,680)" to="(590,680)"/>  <wire from="(360,480)" to="(540,480)"/>  <wire from="(620,680)" to="(680,680)"/>  <wire from="(1180,300)" to="(1180,520)"/>  <wire from="(1120,350)" to="(1410,350)"/>  <wire from="(1390,450)" to="(1390,520)"/>  <wire from="(710,630)" to="(750,630)"/>  <wire from="(1030,250)" to="(1030,340)"/>  <wire from="(1370,540)" to="(1370,670)"/>  <wire from="(210,680)" to="(210,700)"/>  <wire from="(590,700)" to="(880,700)"/>  <wire from="(230,650)" to="(230,740)"/>  <wire from="(970,550)" to="(1080,550)"/>  <wire from="(240,470)" to="(240,500)"/>  <wire from="(1030,250)" to="(1050,250)"/>  <wire from="(1340,530)" to="(1340,580)"/>  <wire from="(650,580)" to="(680,580)"/>  <wire from="(850,660)" to="(1070,660)"/>  <wire from="(160,730)" to="(240,730)"/>  <wire from="(1040,450)" to="(1260,450)"/>  <wire from="(230,740)" to="(240,740)"/>  <wire from="(920,350)" to="(1120,350)"/>  <wire from="(940,250)" to="(950,250)"/>  <wire from="(1040,270)" to="(1050,270)"/>  <wire from="(880,670)" to="(890,670)"/>  <wire from="(540,620)" to="(680,620)"/>  <wire from="(760,230)" to="(770,230)"/>  <wire from="(770,650)" to="(770,660)"/>  <wire from="(760,640)" to="(760,650)"/>  <wire from="(720,270)" to="(720,600)"/>  <wire from="(540,480)" to="(540,620)"/>  <wire from="(850,650)" to="(850,660)"/>  <wire from="(290,250)" to="(290,260)"/>  <wire from="(710,660)" to="(770,660)"/>  <wire from="(970,570)" to="(1260,570)"/>  <wire from="(350,470)" to="(350,490)"/>  <wire from="(790,430)" to="(790,460)"/>  <wire from="(1260,450)" to="(1260,520)"/>  <wire from="(810,610)" to="(810,630)"/>  <wire from="(460,430)" to="(460,650)"/>  <wire from="(920,270)" to="(920,350)"/>  <wire from="(930,270)" to="(930,670)"/>  <wire from="(470,660)" to="(680,660)"/>  <wire from="(910,570)" to="(940,570)"/>  <wire from="(840,390)" to="(1000,390)"/>  <wire from="(670,720)" to="(760,720)"/>  <wire from="(170,720)" to="(240,720)"/>  <wire from="(1010,610)" to="(1020,610)"/>  <wire from="(900,260)" to="(910,260)"/>  <wire from="(820,400)" to="(900,400)"/>  <wire from="(670,670)" to="(680,670)"/>  <wire from="(340,270)" to="(720,270)"/>  <wire from="(550,130)" to="(550,590)"/>  <wire from="(840,450)" to="(840,460)"/>  <wire from="(840,610)" to="(840,620)"/>  <wire from="(910,630)" to="(910,700)"/>  <wire from="(190,190)" to="(630,190)"/>  <wire from="(710,180)" to="(710,590)"/>  <wire from="(820,430)" to="(820,460)"/>  <wire from="(930,670)" to="(1160,670)"/>  <wire from="(870,250)" to="(910,250)"/>  <wire from="(890,670)" to="(930,670)"/>  <wire from="(230,400)" to="(230,420)"/>  <wire from="(830,440)" to="(830,460)"/>  <wire from="(810,740)" to="(810,760)"/>  <wire from="(1030,540)" to="(1200,540)"/>  <wire from="(200,130)" to="(200,160)"/>  <wire from="(760,210)" to="(760,230)"/>  <wire from="(1200,250)" to="(1200,300)"/>  <wire from="(820,340)" to="(1030,340)"/>  <wire from="(710,670)" to="(730,670)"/>  <wire from="(820,490)" to="(1040,490)"/>  <wire from="(290,410)" to="(310,410)"/>  <wire from="(980,360)" to="(980,590)"/>  <wire from="(760,640)" to="(830,640)"/>  <wire from="(230,420)" to="(310,420)"/>  <wire from="(760,250)" to="(770,250)"/>  <wire from="(710,600)" to="(720,600)"/>  <wire from="(860,310)" to="(860,360)"/>  <wire from="(620,680)" to="(620,690)"/>  <wire from="(1010,430)" to="(1010,440)"/>  <wire from="(850,270)" to="(850,280)"/>  <wire from="(820,480)" to="(820,490)"/>  <wire from="(970,540)" to="(1030,540)"/>  <wire from="(950,250)" to="(950,510)"/>  <wire from="(650,170)" to="(650,580)"/>  <wire from="(1210,550)" to="(1210,560)"/>  <wire from="(850,430)" to="(850,450)"/>  <wire from="(810,660)" to="(810,740)"/>  <wire from="(210,430)" to="(210,460)"/>  <wire from="(770,670)" to="(770,690)"/>  <wire from="(200,130)" to="(550,130)"/>  <wire from="(1380,520)" to="(1390,520)"/>  <wire from="(1190,250)" to="(1200,250)"/>  <wire from="(900,400)" to="(900,440)"/>  <wire from="(950,210)" to="(950,250)"/>  <wire from="(260,730)" to="(730,730)"/>  <wire from="(790,610)" to="(790,670)"/>  <wire from="(190,130)" to="(200,130)"/>  <wire from="(330,420)" to="(730,420)"/>  <wire from="(970,590)" to="(980,590)"/>  <wire from="(750,680)" to="(870,680)"/>  <wire from="(1000,390)" to="(1000,400)"/>  <wire from="(210,430)" to="(460,430)"/>  <wire from="(190,320)" to="(500,320)"/>  <wire from="(290,400)" to="(290,410)"/>  <wire from="(1070,660)" to="(1360,660)"/>  <wire from="(360,470)" to="(360,480)"/>  <wire from="(350,170)" to="(650,170)"/>  <wire from="(800,430)" to="(800,460)"/>  <wire from="(880,670)" to="(880,700)"/>  <wire from="(170,630)" to="(170,720)"/>  <wire from="(900,440)" to="(1010,440)"/>  <wire from="(970,560)" to="(1210,560)"/>  <wire from="(1340,530)" to="(1350,530)"/>  <wire from="(230,180)" to="(710,180)"/>  <wire from="(230,270)" to="(320,270)"/>  <wire from="(290,260)" to="(320,260)"/>  <wire from="(710,610)" to="(730,610)"/>  <wire from="(770,670)" to="(790,670)"/>  <wire from="(1360,540)" to="(1360,660)"/>  <wire from="(730,420)" to="(730,610)"/>  <wire from="(1070,620)" to="(1070,660)"/>  <wire from="(1180,520)" to="(1200,520)"/>  <wire from="(1260,520)" to="(1280,520)"/>  <comp lib="5" loc="(190,280)" name="Button">  <a name="label" val="nextmemoryadress"/>  <a name="labelloc" val="west"/>  </comp>  <comp lib="5" loc="(140,700)" name="Button">  <a name="label" val="resetmemory"/>  <a name="labelloc" val="west"/>  </comp>  <comp lib="0" loc="(140,670)" name="Clock">  <a name="label" val="isloading"/>  </comp>  <comp lib="0" loc="(190,130)" name="Clock">  <a name="label" val="loaddata/adress"/>  </comp>  <comp lib="5" loc="(190,320)" name="Button">  <a name="label" val="previousmemoryadress"/>  <a name="labelloc" val="west"/>  </comp>  <comp lib="6" loc="(140,227)" name="Text">  <a name="text" val="loadeddata/adress"/>  </comp>  <comp lib="0" loc="(340,270)" name="Splitter">  <a name="facing" val="west"/>  <a name="incoming" val="8"/>  <a name="appear" val="center"/>  <a name="bit1" val="0"/>  <a name="bit2" val="0"/>  <a name="bit3" val="0"/>  <a name="bit4" val="1"/>  <a name="bit5" val="1"/>  <a name="bit6" val="1"/>  <a name="bit7" val="1"/>  </comp>  <comp lib="0" loc="(810,760)" name="Constant">  <a name="facing" val="north"/>  <a name="value" val="0x0"/>  </comp>  <comp lib="5" loc="(190,190)" name="Button">  <a name="label" val="clearmistakedata/adress"/>  <a name="labelloc" val="west"/>  </comp>  <comp lib="1" loc="(770,690)" name="OR Gate">  <a name="facing" val="north"/>  <a name="size" val="30"/>  <a name="inputs" val="2"/>  </comp>  <comp lib="6" loc="(525,444)" name="Text"/>  <comp lib="0" loc="(760,250)" name="Constant">  <a name="width" val="8"/>  </comp>  <comp lib="0" loc="(910,700)" name="Clock">  <a name="facing" val="north"/>  <a name="label" val="Clock"/>  <a name="labelloc" val="south"/>  </comp>  <comp loc="(1310,520)" name="ALU">  <a name="label" val="ALU"/>  </comp>  <comp lib="6" loc="(165,461)" name="Text">  <a name="text" val="istructiontoload"/>  </comp>  <comp lib="5" loc="(290,250)" name="Hex Digit Display"/>  <comp lib="5" loc="(210,160)" name="Keyboard"/>  <comp lib="5" loc="(230,400)" name="Hex Digit Display"/>  <comp lib="5" loc="(220,460)" name="Keyboard"/>  <comp lib="6" loc="(511,511)" name="Text"/>  <comp lib="6" loc="(109,160)" name="Text">  <a name="text" val="data/adresstoload"/>  </comp>  <comp lib="6" loc="(147,568)" name="Text">  <a name="text" val="loadedinstruction"/>  </comp>  <comp lib="3" loc="(810,240)" name="Adder"/>  <comp lib="1" loc="(620,690)" name="OR Gate">  <a name="size" val="30"/>  <a name="inputs" val="2"/>  </comp>  <comp lib="5" loc="(880,700)" name="Button">  <a name="facing" val="north"/>  <a name="label" val="Reset"/>  </comp>  <comp lib="1" loc="(1050,620)" name="AND Gate">  <a name="size" val="30"/>  <a name="inputs" val="2"/>  </comp>  <comp lib="4" loc="(910,570)" name="RAM">  <a name="dataWidth" val="16"/>  <a name="bus" val="separate"/>  </comp>  <comp lib="6" loc="(837,524)" name="Text">  <a name="text" val="Program memory"/>  </comp>  <comp lib="4" loc="(940,250)" name="Register">  <a name="label" val="PC"/>  </comp>  <comp lib="0" loc="(190,430)" name="Clock">  <a name="label" val="loadinstruction"/>  </comp>  <comp lib="1" loc="(880,620)" name="OR Gate">  <a name="facing" val="north"/>  <a name="size" val="30"/>  <a name="inputs" val="2"/>  </comp>  <comp lib="6" loc="(752,284)" name="Text">  <a name="text" val="Add 1"/>  </comp>  <comp lib="0" loc="(330,420)" name="Splitter">  <a name="facing" val="west"/>  <a name="incoming" val="8"/>  <a name="appear" val="center"/>  <a name="bit1" val="0"/>  <a name="bit2" val="0"/>  <a name="bit3" val="0"/>  <a name="bit4" val="1"/>  <a name="bit5" val="1"/>  <a name="bit6" val="1"/>  <a name="bit7" val="1"/>  </comp>  <comp lib="1" loc="(810,630)" name="Controlled Inverter">  <a name="facing" val="north"/>  <a name="control" val="left"/>  </comp>  <comp lib="1" loc="(770,570)" name="OR Gate">  <a name="width" val="8"/>  <a name="size" val="30"/>  <a name="inputs" val="2"/>  </comp>  <comp lib="6" loc="(1337,445)" name="Text">  <a name="text" val="Output from ACC"/>  </comp>  <comp loc="(710,590)" name="Input"/>  <comp lib="2" loc="(870,250)" name="Multiplexer">  <a name="width" val="8"/>  </comp>  <comp lib="0" loc="(900,260)" name="Constant"/>  <comp lib="4" loc="(1380,520)" name="Register">  <a name="label" val="ACC"/>  </comp>  <comp lib="5" loc="(290,400)" name="Hex Digit Display"/>  <comp loc="(970,540)" name="Instruction Decoding Unit">  <a name="label" val="Instruction Decoder"/>  <a name="labelloc" val="south"/>  </comp>  <comp lib="5" loc="(230,250)" name="Hex Digit Display"/>  <comp lib="1" loc="(840,620)" name="OR Gate">  <a name="facing" val="north"/>  <a name="size" val="30"/>  <a name="inputs" val="2"/>  </comp>  <comp lib="6" loc="(803,171)" name="Text">  <a name="text" val="Student id: 21330134"/>  </comp>  <comp lib="6" loc="(1210,505)" name="Text">  <a name="text" val="SELECT"/>  </comp>  <comp lib="1" loc="(1000,400)" name="AND Gate">  <a name="facing" val="north"/>  <a name="size" val="30"/>  <a name="inputs" val="2"/>  </comp>  <comp lib="1" loc="(820,400)" name="OR Gate">  <a name="facing" val="north"/>  <a name="size" val="30"/>  <a name="inputs" val="8"/>  </comp>  <comp lib="0" loc="(260,730)" name="Splitter">  <a name="facing" val="west"/>  <a name="fanout" val="3"/>  <a name="incoming" val="9"/>  <a name="appear" val="center"/>  <a name="bit1" val="0"/>  <a name="bit2" val="0"/>  <a name="bit3" val="0"/>  <a name="bit4" val="0"/>  <a name="bit5" val="0"/>  <a name="bit6" val="0"/>  <a name="bit7" val="1"/>  <a name="bit8" val="2"/>  </comp>  <comp lib="5" loc="(200,500)" name="Button">  <a name="label" val="clearmistakeinstruction"/>  <a name="labelloc" val="west"/>  </comp>  <comp lib="2" loc="(1230,530)" name="Multiplexer">  <a name="width" val="8"/>  </comp>  <comp lib="1" loc="(850,280)" name="OR Gate">  <a name="facing" val="north"/>  <a name="size" val="30"/>  <a name="inputs" val="2"/>  </comp>  <comp lib="0" loc="(820,480)" name="Splitter">  <a name="facing" val="north"/>  <a name="fanout" val="8"/>  <a name="incoming" val="8"/>  <a name="appear" val="center"/>  </comp>  <comp lib="6" loc="(1252,316)" name="Text">  <a name="text" val="Output from data memory"/>  </comp>  <comp lib="1" loc="(1010,610)" name="NOT Gate"/>  <comp lib="6" loc="(1230,239)" name="Text">  <a name="text" val="Data memory"/>  </comp>  <comp lib="5" loc="(210,640)" name="TTY"/>  <comp lib="4" loc="(1190,250)" name="RAM">  <a name="bus" val="separate"/>  </comp>  </circuit>  <circuit name="ALU">  <a name="circuit" val="ALU"/>  <a name="clabel" val=""/>  <a name="clabelup" val="east"/>  <a name="clabelfont" val="SansSerif plain 12"/>  <wire from="(470,140)" to="(530,140)"/>  <wire from="(150,200)" to="(150,270)"/>  <wire from="(470,190)" to="(520,190)"/>  <wire from="(460,240)" to="(510,240)"/>  <wire from="(520,270)" to="(580,270)"/>  <wire from="(500,280)" to="(500,290)"/>  <wire from="(540,250)" to="(580,250)"/>  <wire from="(150,270)" to="(150,290)"/>  <wire from="(520,190)" to="(520,270)"/>  <wire from="(390,80)" to="(430,80)"/>  <wire from="(390,180)" to="(430,180)"/>  <wire from="(390,240)" to="(430,240)"/>  <wire from="(490,300)" to="(580,300)"/>  <wire from="(150,100)" to="(430,100)"/>  <wire from="(150,200)" to="(430,200)"/>  <wire from="(150,50)" to="(150,100)"/>  <wire from="(530,140)" to="(530,260)"/>  <wire from="(150,150)" to="(150,200)"/>  <wire from="(480,340)" to="(490,340)"/>  <wire from="(550,50)" to="(550,240)"/>  <wire from="(600,400)" to="(600,590)"/>  <wire from="(390,130)" to="(390,180)"/>  <wire from="(390,270)" to="(390,330)"/>  <wire from="(510,280)" to="(580,280)"/>  <wire from="(130,80)" to="(390,80)"/>  <wire from="(150,290)" to="(150,350)"/>  <wire from="(530,260)" to="(580,260)"/>  <wire from="(620,320)" to="(680,320)"/>  <wire from="(390,240)" to="(390,270)"/>  <wire from="(390,130)" to="(430,130)"/>  <wire from="(390,270)" to="(430,270)"/>  <wire from="(120,590)" to="(600,590)"/>  <wire from="(470,280)" to="(500,280)"/>  <wire from="(540,90)" to="(540,250)"/>  <wire from="(550,240)" to="(580,240)"/>  <wire from="(150,150)" to="(430,150)"/>  <wire from="(150,290)" to="(430,290)"/>  <wire from="(390,330)" to="(410,330)"/>  <wire from="(130,270)" to="(150,270)"/>  <wire from="(510,240)" to="(510,280)"/>  <wire from="(490,300)" to="(490,340)"/>  <wire from="(150,100)" to="(150,150)"/>  <wire from="(150,50)" to="(550,50)"/>  <wire from="(90,510)" to="(680,510)"/>  <wire from="(390,80)" to="(390,130)"/>  <wire from="(500,290)" to="(580,290)"/>  <wire from="(390,180)" to="(390,240)"/>  <wire from="(470,90)" to="(540,90)"/>  <wire from="(150,350)" to="(410,350)"/>  <comp lib="0" loc="(680,510)" name="Pin">  <a name="facing" val="west"/>  <a name="output" val="true"/>  <a name="width" val="8"/>  <a name="label" val="Not Equal Zero"/>  <a name="labelloc" val="east"/>  </comp>  <comp lib="1" loc="(460,240)" name="NOT Gate">  <a name="width" val="8"/>  </comp>  <comp lib="0" loc="(130,80)" name="Pin">  <a name="width" val="8"/>  <a name="tristate" val="false"/>  <a name="label" val="ACC to ALU"/>  <a name="labelloc" val="south"/>  </comp>  <comp lib="3" loc="(470,340)" name="Divider"/>  <comp lib="3" loc="(470,90)" name="Adder"/>  <comp lib="0" loc="(90,510)" name="Constant">  <a name="width" val="8"/>  <a name="value" val="0x0"/>  </comp>  <comp lib="1" loc="(470,140)" name="XOR Gate">  <a name="width" val="8"/>  <a name="size" val="30"/>  <a name="inputs" val="2"/>  </comp>  <comp lib="0" loc="(130,270)" name="Pin">  <a name="width" val="8"/>  <a name="tristate" val="false"/>  <a name="label" val="Inp to ALU"/>  <a name="labelloc" val="south"/>  </comp>  <comp lib="0" loc="(120,590)" name="Pin">  <a name="width" val="4"/>  <a name="tristate" val="false"/>  <a name="label" val="ALU Operation"/>  </comp>  <comp lib="3" loc="(470,280)" name="Subtractor"/>  <comp lib="2" loc="(620,320)" name="Multiplexer">  <a name="select" val="4"/>  <a name="width" val="8"/>  </comp>  <comp lib="0" loc="(680,320)" name="Pin">  <a name="facing" val="west"/>  <a name="output" val="true"/>  <a name="width" val="8"/>  <a name="label" val="ALU out to ACC"/>  <a name="labelloc" val="south"/>  </comp>  <comp lib="3" loc="(470,190)" name="Multiplier"/>  </circuit>  <circuit name="Instruction Decoding Unit">  <a name="circuit" val="Instruction Decoding Unit"/>  <a name="clabel" val=""/>  <a name="clabelup" val="east"/>  <a name="clabelfont" val="SansSerif plain 12"/>  <wire from="(330,110)" to="(580,110)"/>  <wire from="(530,250)" to="(580,250)"/>  <wire from="(250,250)" to="(310,250)"/>  <wire from="(330,130)" to="(520,130)"/>  <wire from="(250,60)" to="(560,60)"/>  <wire from="(530,120)" to="(530,250)"/>  <wire from="(330,90)" to="(380,90)"/>  <wire from="(520,290)" to="(580,290)"/>  <wire from="(310,240)" to="(310,250)"/>  <wire from="(380,80)" to="(380,90)"/>  <wire from="(540,210)" to="(580,210)"/>  <wire from="(250,170)" to="(250,250)"/>  <wire from="(350,540)" to="(590,540)"/>  <wire from="(380,80)" to="(490,80)"/>  <wire from="(200,150)" to="(230,150)"/>  <wire from="(560,170)" to="(580,170)"/>  <wire from="(330,100)" to="(490,100)"/>  <wire from="(560,60)" to="(560,170)"/>  <wire from="(250,60)" to="(250,160)"/>  <wire from="(520,130)" to="(520,290)"/>  <wire from="(330,230)" to="(350,230)"/>  <wire from="(520,90)" to="(540,90)"/>  <wire from="(230,40)" to="(230,150)"/>  <wire from="(140,160)" to="(160,160)"/>  <wire from="(230,40)" to="(250,40)"/>  <wire from="(540,90)" to="(540,210)"/>  <wire from="(350,230)" to="(350,540)"/>  <wire from="(380,90)" to="(380,140)"/>  <wire from="(180,160)" to="(250,160)"/>  <wire from="(180,170)" to="(250,170)"/>  <wire from="(330,120)" to="(530,120)"/>  <wire from="(380,140)" to="(580,140)"/>  <comp lib="1" loc="(520,90)" name="OR Gate">  <a name="size" val="30"/>  <a name="inputs" val="2"/>  </comp>  <comp lib="0" loc="(580,170)" name="Pin">  <a name="facing" val="west"/>  <a name="output" val="true"/>  <a name="width" val="4"/>  <a name="label" val="ALU Operation"/>  <a name="labelloc" val="east"/>  </comp>  <comp lib="0" loc="(140,160)" name="Pin">  <a name="width" val="16"/>  <a name="tristate" val="false"/>  <a name="label" val="Instruction"/>  </comp>  <comp lib="0" loc="(580,250)" name="Pin">  <a name="facing" val="west"/>  <a name="output" val="true"/>  <a name="label" val="Branch always"/>  <a name="labelloc" val="east"/>  </comp>  <comp lib="2" loc="(310,240)" name="Decoder">  <a name="select" val="4"/>  </comp>  <comp lib="0" loc="(580,110)" name="Pin">  <a name="facing" val="west"/>  <a name="output" val="true"/>  <a name="label" val="Write Memory"/>  <a name="labelloc" val="east"/>  </comp>  <comp lib="0" loc="(160,160)" name="Splitter">  <a name="fanout" val="4"/>  <a name="incoming" val="16"/>  <a name="appear" val="center"/>  <a name="bit1" val="0"/>  <a name="bit2" val="0"/>  <a name="bit3" val="0"/>  <a name="bit4" val="1"/>  <a name="bit5" val="1"/>  <a name="bit6" val="1"/>  <a name="bit7" val="1"/>  <a name="bit8" val="2"/>  <a name="bit9" val="2"/>  <a name="bit10" val="2"/>  <a name="bit11" val="2"/>  <a name="bit12" val="3"/>  <a name="bit13" val="3"/>  <a name="bit14" val="3"/>  <a name="bit15" val="3"/>  </comp>  <comp lib="0" loc="(580,140)" name="Pin">  <a name="facing" val="west"/>  <a name="output" val="true"/>  <a name="label" val="Use Immediate"/>  <a name="labelloc" val="east"/>  </comp>  <comp lib="0" loc="(250,40)" name="Pin">  <a name="facing" val="west"/>  <a name="output" val="true"/>  <a name="width" val="8"/>  <a name="label" val="Address or Immediate"/>  <a name="labelloc" val="east"/>  </comp>  <comp lib="0" loc="(580,210)" name="Pin">  <a name="facing" val="west"/>  <a name="output" val="true"/>  <a name="label" val="Write ACC"/>  <a name="labelloc" val="east"/>  </comp>  <comp lib="0" loc="(200,150)" name="Splitter">  <a name="facing" val="west"/>  <a name="incoming" val="8"/>  <a name="appear" val="center"/>  <a name="bit1" val="0"/>  <a name="bit2" val="0"/>  <a name="bit3" val="0"/>  <a name="bit4" val="1"/>  <a name="bit5" val="1"/>  <a name="bit6" val="1"/>  <a name="bit7" val="1"/>  </comp>  <comp lib="0" loc="(580,290)" name="Pin">  <a name="facing" val="west"/>  <a name="output" val="true"/>  <a name="label" val="Branch NE zero"/>  <a name="labelloc" val="east"/>  </comp>  <comp lib="0" loc="(590,540)" name="Pin">  <a name="facing" val="west"/>  <a name="output" val="true"/>  <a name="label" val="Halt"/>  <a name="labelloc" val="east"/>  </comp>  </circuit>  <circuit name="Input">  <a name="circuit" val="Input"/>  <a name="clabel" val=""/>  <a name="clabelup" val="west"/>  <a name="clabelfont" val="SansSerif plain 12"/>  <wire from="(720,190)" to="(720,260)"/>  <wire from="(950,670)" to="(1140,670)"/>  <wire from="(700,310)" to="(740,310)"/>  <wire from="(760,850)" to="(800,850)"/>  <wire from="(490,410)" to="(490,430)"/>  <wire from="(650,810)" to="(650,840)"/>  <wire from="(540,460)" to="(540,490)"/>  <wire from="(440,200)" to="(440,230)"/>  <wire from="(630,300)" to="(740,300)"/>  <wire from="(860,570)" to="(880,570)"/>  <wire from="(500,440)" to="(500,860)"/>  <wire from="(760,310)" to="(780,310)"/>  <wire from="(720,550)" to="(880,550)"/>  <wire from="(430,230)" to="(440,230)"/>  <wire from="(540,460)" to="(740,460)"/>  <wire from="(1170,710)" to="(1200,710)"/>  <wire from="(540,140)" to="(540,200)"/>  <wire from="(700,220)" to="(700,280)"/>  <wire from="(790,670)" to="(800,670)"/>  <wire from="(670,550)" to="(680,550)"/>  <wire from="(280,520)" to="(480,520)"/>  <wire from="(580,220)" to="(590,220)"/>  <wire from="(760,650)" to="(760,660)"/>  <wire from="(660,150)" to="(710,150)"/>  <wire from="(540,500)" to="(540,820)"/>  <wire from="(530,470)" to="(760,470)"/>  <wire from="(1060,460)" to="(1120,460)"/>  <wire from="(830,430)" to="(830,520)"/>  <wire from="(1200,610)" to="(1200,620)"/>  <wire from="(490,190)" to="(720,190)"/>  <wire from="(670,240)" to="(670,260)"/>  <wire from="(690,500)" to="(690,520)"/>  <wire from="(1010,610)" to="(1030,610)"/>  <wire from="(860,460)" to="(860,570)"/>  <wire from="(790,600)" to="(790,640)"/>  <wire from="(540,140)" to="(560,140)"/>  <wire from="(480,470)" to="(510,470)"/>  <wire from="(820,420)" to="(820,520)"/>  <wire from="(710,860)" to="(740,860)"/>  <wire from="(520,200)" to="(540,200)"/>  <wire from="(430,200)" to="(440,200)"/>  <wire from="(630,290)" to="(1150,290)"/>  <wire from="(1060,570)" to="(1090,570)"/>  <wire from="(760,650)" to="(960,650)"/>  <wire from="(610,850)" to="(680,850)"/>  <wire from="(690,520)" to="(820,520)"/>  <wire from="(680,510)" to="(810,510)"/>  <wire from="(820,420)" to="(890,420)"/>  <wire from="(920,690)" to="(930,690)"/>  <wire from="(1120,610)" to="(1120,650)"/>  <wire from="(650,220)" to="(650,270)"/>  <wire from="(510,470)" to="(510,850)"/>  <wire from="(1070,590)" to="(1090,590)"/>  <wire from="(690,220)" to="(700,220)"/>  <wire from="(280,490)" to="(480,490)"/>  <wire from="(490,410)" to="(560,410)"/>  <wire from="(760,470)" to="(760,650)"/>  <wire from="(740,680)" to="(740,690)"/>  <wire from="(590,370)" to="(590,380)"/>  <wire from="(800,660)" to="(800,670)"/>  <wire from="(800,90)" to="(860,90)"/>  <wire from="(810,510)" to="(910,510)"/>  <wire from="(590,380)" to="(630,380)"/>  <wire from="(1070,520)" to="(1130,520)"/>  <wire from="(520,230)" to="(520,440)"/>  <wire from="(870,340)" to="(910,340)"/>  <wire from="(830,160)" to="(850,160)"/>  <wire from="(640,820)" to="(640,860)"/>  <wire from="(1060,520)" to="(1060,570)"/>  <wire from="(740,200)" to="(740,240)"/>  <wire from="(840,620)" to="(840,660)"/>  <wire from="(490,130)" to="(520,130)"/>  <wire from="(1160,280)" to="(1160,460)"/>  <wire from="(790,250)" to="(940,250)"/>  <wire from="(650,840)" to="(740,840)"/>  <wire from="(470,530)" to="(480,530)"/>  <wire from="(710,90)" to="(710,150)"/>  <wire from="(500,860)" to="(580,860)"/>  <wire from="(850,110)" to="(860,110)"/>  <wire from="(280,460)" to="(480,460)"/>  <wire from="(970,560)" to="(1020,560)"/>  <wire from="(700,850)" to="(700,860)"/>  <wire from="(710,860)" to="(710,870)"/>  <wire from="(930,680)" to="(930,690)"/>  <wire from="(670,260)" to="(720,260)"/>  <wire from="(1200,710)" to="(1200,730)"/>  <wire from="(490,400)" to="(490,410)"/>  <wire from="(640,860)" to="(700,860)"/>  <wire from="(1060,490)" to="(1060,520)"/>  <wire from="(940,160)" to="(940,250)"/>  <wire from="(650,270)" to="(950,270)"/>  <wire from="(940,560)" to="(940,640)"/>  <wire from="(440,200)" to="(480,200)"/>  <wire from="(680,830)" to="(680,850)"/>  <wire from="(970,560)" to="(970,600)"/>  <wire from="(640,820)" to="(660,820)"/>  <wire from="(740,680)" to="(760,680)"/>  <wire from="(1060,490)" to="(1260,490)"/>  <wire from="(500,440)" to="(520,440)"/>  <wire from="(920,210)" to="(950,210)"/>  <wire from="(1040,620)" to="(1200,620)"/>  <wire from="(470,500)" to="(480,500)"/>  <wire from="(510,850)" to="(580,850)"/>  <wire from="(970,600)" to="(980,600)"/>  <wire from="(1130,480)" to="(1130,520)"/>  <wire from="(830,630)" to="(1160,630)"/>  <wire from="(700,90)" to="(710,90)"/>  <wire from="(280,430)" to="(480,430)"/>  <wire from="(780,170)" to="(790,170)"/>  <wire from="(540,490)" to="(550,490)"/>  <wire from="(660,140)" to="(660,150)"/>  <wire from="(580,280)" to="(700,280)"/>  <wire from="(680,240)" to="(680,250)"/>  <wire from="(1200,630)" to="(1200,710)"/>  <wire from="(750,530)" to="(750,670)"/>  <wire from="(1080,650)" to="(1120,650)"/>  <wire from="(510,230)" to="(510,240)"/>  <wire from="(480,520)" to="(480,530)"/>  <wire from="(490,370)" to="(490,380)"/>  <wire from="(630,220)" to="(630,290)"/>  <wire from="(970,530)" to="(970,560)"/>  <wire from="(540,820)" to="(640,820)"/>  <wire from="(830,520)" to="(1060,520)"/>  <wire from="(630,300)" to="(630,380)"/>  <wire from="(1130,450)" to="(1130,460)"/>  <wire from="(1030,590)" to="(1030,610)"/>  <wire from="(600,240)" to="(600,260)"/>  <wire from="(480,530)" to="(520,530)"/>  <wire from="(1050,570)" to="(1060,570)"/>  <wire from="(810,510)" to="(810,670)"/>  <wire from="(740,630)" to="(830,630)"/>  <wire from="(790,210)" to="(820,210)"/>  <wire from="(750,530)" to="(970,530)"/>  <wire from="(480,200)" to="(480,240)"/>  <wire from="(1150,290)" to="(1150,450)"/>  <wire from="(470,470)" to="(480,470)"/>  <wire from="(580,220)" to="(580,280)"/>  <wire from="(1160,650)" to="(1250,650)"/>  <wire from="(490,130)" to="(490,190)"/>  <wire from="(650,810)" to="(660,810)"/>  <wire from="(750,670)" to="(760,670)"/>  <wire from="(700,280)" to="(1160,280)"/>  <wire from="(620,220)" to="(630,220)"/>  <wire from="(890,370)" to="(890,420)"/>  <wire from="(1160,610)" to="(1160,630)"/>  <wire from="(520,530)" to="(570,530)"/>  <wire from="(440,130)" to="(490,130)"/>  <wire from="(860,460)" to="(920,460)"/>  <wire from="(520,840)" to="(580,840)"/>  <wire from="(440,370)" to="(440,380)"/>  <wire from="(480,490)" to="(480,500)"/>  <wire from="(800,660)" to="(840,660)"/>  <wire from="(670,520)" to="(670,550)"/>  <wire from="(570,500)" to="(570,530)"/>  <wire from="(440,130)" to="(440,160)"/>  <wire from="(680,850)" to="(680,870)"/>  <wire from="(440,380)" to="(480,380)"/>  <wire from="(920,160)" to="(940,160)"/>  <wire from="(920,560)" to="(940,560)"/>  <wire from="(790,640)" to="(940,640)"/>  <wire from="(910,340)" to="(910,510)"/>  <wire from="(680,870)" to="(710,870)"/>  <wire from="(510,470)" to="(530,470)"/>  <wire from="(480,440)" to="(500,440)"/>  <wire from="(1130,520)" to="(1260,520)"/>  <wire from="(430,160)" to="(440,160)"/>  <wire from="(470,440)" to="(480,440)"/>  <wire from="(610,250)" to="(680,250)"/>  <wire from="(530,230)" to="(530,470)"/>  <wire from="(1140,460)" to="(1160,460)"/>  <wire from="(1130,450)" to="(1150,450)"/>  <wire from="(650,220)" to="(660,220)"/>  <wire from="(680,500)" to="(680,510)"/>  <wire from="(740,240)" to="(740,250)"/>  <wire from="(630,290)" to="(630,300)"/>  <wire from="(1200,630)" to="(1240,630)"/>  <wire from="(700,310)" to="(700,380)"/>  <wire from="(610,550)" to="(610,810)"/>  <wire from="(480,460)" to="(480,470)"/>  <wire from="(680,250)" to="(740,250)"/>  <wire from="(960,620)" to="(960,650)"/>  <wire from="(750,440)" to="(750,530)"/>  <wire from="(700,850)" to="(740,850)"/>  <wire from="(700,280)" to="(700,310)"/>  <wire from="(1070,520)" to="(1070,590)"/>  <wire from="(740,240)" to="(840,240)"/>  <wire from="(1200,620)" to="(1200,630)"/>  <wire from="(740,460)" to="(740,630)"/>  <wire from="(770,190)" to="(770,230)"/>  <wire from="(790,210)" to="(790,250)"/>  <wire from="(480,240)" to="(510,240)"/>  <wire from="(670,520)" to="(690,520)"/>  <wire from="(490,430)" to="(830,430)"/>  <wire from="(850,210)" to="(880,210)"/>  <wire from="(430,130)" to="(440,130)"/>  <wire from="(540,200)" to="(740,200)"/>  <wire from="(910,90)" to="(910,150)"/>  <wire from="(950,210)" to="(950,270)"/>  <wire from="(710,150)" to="(790,150)"/>  <wire from="(850,110)" to="(850,160)"/>  <wire from="(660,370)" to="(660,380)"/>  <wire from="(850,160)" to="(900,160)"/>  <wire from="(540,490)" to="(540,500)"/>  <wire from="(610,240)" to="(610,250)"/>  <wire from="(480,500)" to="(540,500)"/>  <wire from="(720,190)" to="(770,190)"/>  <wire from="(1160,630)" to="(1160,650)"/>  <wire from="(840,230)" to="(840,240)"/>  <wire from="(830,620)" to="(830,630)"/>  <wire from="(610,550)" to="(670,550)"/>  <wire from="(480,430)" to="(480,440)"/>  <wire from="(1040,590)" to="(1040,620)"/>  <wire from="(770,230)" to="(830,230)"/>  <wire from="(610,810)" to="(650,810)"/>  <wire from="(520,440)" to="(750,440)"/>  <wire from="(660,380)" to="(700,380)"/>  <wire from="(860,570)" to="(860,600)"/>  <wire from="(1140,670)" to="(1260,670)"/>  <wire from="(520,690)" to="(520,840)"/>  <wire from="(810,670)" to="(920,670)"/>  <wire from="(740,690)" to="(890,690)"/>  <wire from="(870,370)" to="(890,370)"/>  <wire from="(960,620)" to="(980,620)"/>  <wire from="(1000,580)" to="(1020,580)"/>  <wire from="(520,530)" to="(520,690)"/>  <wire from="(520,690)" to="(740,690)"/>  <wire from="(1140,610)" to="(1140,670)"/>  <wire from="(790,600)" to="(820,600)"/>  <wire from="(600,260)" to="(670,260)"/>  <wire from="(900,90)" to="(910,90)"/>  <wire from="(850,600)" to="(860,600)"/>  <comp lib="5" loc="(490,370)" name="Hex Digit Display"/>  <comp lib="0" loc="(280,490)" name="Pin">  <a name="tristate" val="false"/>  <a name="label" val="loadinstruction"/>  </comp>  <comp lib="0" loc="(1130,480)" name="Splitter">  <a name="facing" val="north"/>  <a name="fanout" val="3"/>  <a name="incoming" val="16"/>  <a name="appear" val="center"/>  <a name="bit1" val="0"/>  <a name="bit2" val="0"/>  <a name="bit3" val="0"/>  <a name="bit4" val="1"/>  <a name="bit5" val="1"/>  <a name="bit6" val="1"/>  <a name="bit7" val="1"/>  <a name="bit8" val="2"/>  <a name="bit9" val="2"/>  <a name="bit10" val="2"/>  <a name="bit11" val="2"/>  <a name="bit12" val="2"/>  <a name="bit13" val="2"/>  <a name="bit14" val="2"/>  <a name="bit15" val="2"/>  </comp>  <comp lib="0" loc="(280,520)" name="Pin">  <a name="tristate" val="false"/>  <a name="label" val="clearmistakeinstruction"/>  </comp>  <comp lib="0" loc="(700,90)" name="Pin">  <a name="width" val="7"/>  <a name="tristate" val="false"/>  <a name="label" val="loaddata/address"/>  </comp>  <comp lib="0" loc="(280,430)" name="Pin">  <a name="tristate" val="false"/>  <a name="label" val="previousmemoryadress"/>  </comp>  <comp lib="0" loc="(920,210)" name="Bit Extender">  <a name="in\_width" val="7"/>  <a name="out\_width" val="4"/>  </comp>  <comp lib="0" loc="(870,340)" name="Pin">  <a name="tristate" val="false"/>  <a name="label" val="instructionhasvalue"/>  </comp>  <comp lib="0" loc="(1260,670)" name="Pin">  <a name="facing" val="west"/>  <a name="output" val="true"/>  <a name="label" val="blockload"/>  <a name="labelloc" val="east"/>  </comp>  <comp lib="5" loc="(660,370)" name="Hex Digit Display"/>  <comp lib="0" loc="(760,850)" name="Splitter">  <a name="facing" val="west"/>  <a name="fanout" val="3"/>  <a name="incoming" val="9"/>  <a name="appear" val="center"/>  <a name="bit1" val="0"/>  <a name="bit2" val="0"/>  <a name="bit3" val="0"/>  <a name="bit4" val="0"/>  <a name="bit5" val="0"/>  <a name="bit6" val="0"/>  <a name="bit7" val="1"/>  <a name="bit8" val="2"/>  </comp>  <comp lib="0" loc="(490,400)" name="Splitter">  <a name="facing" val="north"/>  <a name="incoming" val="8"/>  <a name="appear" val="center"/>  <a name="bit1" val="0"/>  <a name="bit2" val="0"/>  <a name="bit3" val="0"/>  <a name="bit4" val="1"/>  <a name="bit5" val="1"/>  <a name="bit6" val="1"/>  <a name="bit7" val="1"/>  </comp>  <comp lib="1" loc="(610,850)" name="OR Gate">  <a name="size" val="30"/>  <a name="inputs" val="3"/>  </comp>  <comp lib="0" loc="(280,460)" name="Pin">  <a name="tristate" val="false"/>  <a name="label" val="nextmemoryadress"/>  </comp>  <comp lib="0" loc="(1080,650)" name="Pin">  <a name="tristate" val="false"/>  <a name="label" val="isloading"/>  </comp>  <comp lib="3" loc="(830,160)" name="Subtractor">  <a name="width" val="7"/>  </comp>  <comp lib="6" loc="(635,393)" name="Text">  <a name="text" val="data or memory value in base 10"/>  </comp>  <comp lib="0" loc="(560,410)" name="Pin">  <a name="facing" val="west"/>  <a name="output" val="true"/>  <a name="width" val="8"/>  <a name="label" val="memoryaddresscounter"/>  <a name="labelloc" val="east"/>  </comp>  <comp lib="1" loc="(950,670)" name="Controlled Inverter"/>  <comp lib="0" loc="(1260,520)" name="Pin">  <a name="facing" val="west"/>  <a name="output" val="true"/>  <a name="width" val="16"/>  <a name="label" val="datainram"/>  <a name="labelloc" val="east"/>  </comp>  <comp lib="5" loc="(590,370)" name="Hex Digit Display"/>  <comp lib="0" loc="(560,140)" name="Pin">  <a name="facing" val="west"/>  <a name="output" val="true"/>  <a name="label" val="clearmistakedata/addressout"/>  <a name="labelloc" val="east"/>  </comp>  <comp lib="3" loc="(920,560)" name="Adder"/>  <comp lib="1" loc="(790,670)" name="OR Gate">  <a name="size" val="30"/>  <a name="inputs" val="3"/>  </comp>  <comp lib="0" loc="(800,850)" name="Pin">  <a name="facing" val="west"/>  <a name="output" val="true"/>  <a name="width" val="9"/>  <a name="label" val="consoleoutput"/>  <a name="labelloc" val="east"/>  </comp>  <comp lib="4" loc="(850,600)" name="Register"/>  <comp lib="4" loc="(1230,570)" name="RAM">  <a name="dataWidth" val="16"/>  <a name="bus" val="separate"/>  </comp>  <comp lib="1" loc="(920,690)" name="NOT Gate"/>  <comp lib="0" loc="(1260,490)" name="Pin">  <a name="facing" val="west"/>  <a name="output" val="true"/>  <a name="width" val="8"/>  <a name="label" val="adressinram"/>  <a name="labelloc" val="east"/>  </comp>  <comp lib="5" loc="(660,820)" name="TTY"/>  <comp lib="0" loc="(780,310)" name="Pin">  <a name="facing" val="west"/>  <a name="output" val="true"/>  <a name="width" val="8"/>  <a name="label" val="data/adresstoloadcounter"/>  <a name="labelloc" val="east"/>  </comp>  <comp lib="1" loc="(920,160)" name="Controlled Buffer">  <a name="width" val="7"/>  <a name="control" val="left"/>  </comp>  <comp lib="0" loc="(870,370)" name="Pin">  <a name="width" val="7"/>  <a name="tristate" val="false"/>  <a name="label" val="loadinstruction"/>  </comp>  <comp lib="0" loc="(780,170)" name="Constant">  <a name="width" val="7"/>  <a name="value" val="0x30"/>  </comp>  <comp lib="1" loc="(1010,610)" name="OR Gate">  <a name="size" val="30"/>  <a name="inputs" val="2"/>  </comp>  <comp lib="4" loc="(1060,460)" name="ROM">  <a name="contents">addr/data: 8 8  41\*0 21 0 23 8\*0 50 14\*0 26  13\*0 24 7\*0 22 39\*0 10 30\*0 20  4\*0 25 20\*0 40 11\*0 f0  </a>  </comp>  <comp lib="5" loc="(440,370)" name="Hex Digit Display"/>  <comp lib="3" loc="(900,100)" name="Comparator">  <a name="width" val="7"/>  <a name="mode" val="unsigned"/>  </comp>  <comp lib="0" loc="(1000,580)" name="Constant"/>  <comp lib="4" loc="(620,220)" name="Register">  <a name="width" val="4"/>  </comp>  <comp lib="0" loc="(1250,650)" name="Pin">  <a name="facing" val="west"/>  <a name="output" val="true"/>  <a name="label" val="clocktick"/>  <a name="labelloc" val="east"/>  </comp>  <comp lib="0" loc="(800,90)" name="Constant">  <a name="width" val="7"/>  <a name="value" val="0xa"/>  </comp>  <comp lib="0" loc="(720,550)" name="Bit Extender">  <a name="in\_width" val="7"/>  <a name="out\_width" val="8"/>  </comp>  <comp lib="0" loc="(1170,710)" name="Pin">  <a name="tristate" val="false"/>  <a name="label" val="resetmemoryin"/>  </comp>  <comp lib="6" loc="(441,299)" name="Text">  <a name="text" val="current memory adress in hex"/>  </comp>  <comp lib="4" loc="(850,210)" name="Register">  <a name="width" val="7"/>  </comp>  <comp lib="4" loc="(690,220)" name="Register">  <a name="width" val="4"/>  </comp>  <comp lib="0" loc="(430,160)" name="Pin">  <a name="tristate" val="false"/>  <a name="label" val="loaddata/addresstick"/>  </comp>  <comp lib="0" loc="(1240,630)" name="Pin">  <a name="facing" val="west"/>  <a name="output" val="true"/>  <a name="label" val="resetmemoryout"/>  <a name="labelloc" val="east"/>  </comp>  <comp lib="1" loc="(520,200)" name="OR Gate">  <a name="facing" val="north"/>  <a name="size" val="30"/>  <a name="inputs" val="3"/>  </comp>  <comp lib="0" loc="(760,310)" name="Splitter">  <a name="facing" val="west"/>  <a name="incoming" val="8"/>  <a name="appear" val="center"/>  <a name="bit0" val="1"/>  <a name="bit2" val="1"/>  <a name="bit3" val="1"/>  <a name="bit4" val="0"/>  <a name="bit5" val="0"/>  <a name="bit6" val="0"/>  <a name="bit7" val="0"/>  </comp>  <comp lib="4" loc="(1050,570)" name="Counter"/>  <comp lib="0" loc="(430,230)" name="Pin">  <a name="tristate" val="false"/>  <a name="label" val="clearmistakedata/address"/>  </comp>  </circuit>  </project> |
| --- |

# Appendix B: The MIPS Code

| .data  enterMsg1: .asciiz "Please enter the last four digits of your student id \n"  enterMsg2: .asciiz "Press enter between each digit \n"  enterMsg3: .asciiz "Enter next digit \n"  enterMsg4: .asciiz "\nThe total of the digits is: "  enterMsg5: .asciiz "\nThe total of the digits multiplied is: "  enterMsg6: .asciiz "array output: \n"  newtab: .asciiz "\t"  array: .space 4  .text  #sets up loop var  addi $t2, $zero, 4  #sets up $t6 not equal to zero  addi $t6, $zero, 1  # output the initial instruction text to the console  addi $v0, $zero, 4  la $a0, enterMsg1  syscall  ##  # output the next line instruction text to the console  addi $v0, $zero, 4  la $a0, enterMsg2  syscall  askandcollect:  # output the text asking for the next digit to the console  addi $v0, $zero, 4  la $a0, enterMsg3  syscall    # read an integer from keyboard input and store the input in $s0 for the total  addi $v0, $zero, 5  syscall    sb $v0, array($t2)    #takes away 1 from $t2  subi $t2, $t2, 1    bgtz $t2, askandcollect  # outputs a newline  addi $v0, $zero, 4  la $a0, enterMsg6  syscall  #sets up loop var  addi $t2, $zero, 4  outputarrayadd:  #gets position at $t2 and outputs it  lb $t4, array($t2)  move $a0, $t4  addi $v0, $zero, 1  syscall    # outputs a newline  addi $v0, $zero, 4  la $a0, newtab  syscall    #takes away 1 from $t2  subi $t2, $t2, 1    add $t5, $t5, $t4  mul $t6, $t6, $t4    bgtz $t2, outputarrayadd  ##  # output the total intro to the console  addi $v0, $zero, 4  la $a0, enterMsg4  syscall  #output the total of the nums from $t5  move $a0, $t5  addi $v0, $zero, 1  syscall  # output the total timesed intro to the console  addi $v0, $zero, 4  la $a0, enterMsg5  syscall  #output the total of the nums timesed from $t6  move $a0, $t6  addi $v0, $zero, 1  syscall  addi $v0, $zero, 10  syscall |
| --- |

# Appendix C: Completed Marking Grid

| Criteria | 0-29% | 30-39% | 40-49% | 50-59% | 60-69% | 70-85% | 86-100% |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Logisim CPU simulation amendment and testing | There is little or no evidence of using or developing the Logisim CPU simulation.  There is little or no evidence of a professional approach to the tasks. | There is some evidence of trying at least one of the Logisim CPU tasks.  There is little evidence of a professional approach to the tasks. | There is evidence that at least two of the Logisim CPU tasks are complete and working appropriately.  There is some evidence of considering a professional approach to the completed tasks. | There is evidence that at least two of the Logisim CPU tasks are complete and working appropriately with attempts at further tasks.  There is some evidence of applying a professional approach to the completed tasks. | There is evidence that most of the Logisim CPU tasks are complete and working appropriately.  Most of the tasks have been done with a suitably professional approach. | There is evidence that most of the Logisim CPU tasks are complete and working appropriately with an attempt at all tasks.  Most of the tasks have been done with a highly professional approach. | There is evidence that all of the Logisim CPU tasks are complete and working, although there may be issues with some aspects on one or two tasks.  Most of the tasks have been done with a highly professional approach. |
| MIPS Assembly language amendment and testing | There is little or no evidence of using or developing the MIPS Assembly language program.  There is little or no evidence of a professional approach to the tasks. | There is some evidence of trying at least one of the tasks to amend the supplied MIPS Assembly language program.  There is little evidence of a professional approach to the tasks. | There is evidence that at least two of the MIPS Assembly language tasks are complete and working appropriately.  There is some evidence of considering a professional approach to the completed tasks | There is evidence that at least two of the MIPS Assembly language tasks are complete and working appropriately with attempts at further tasks.  There is some evidence of applying a professional approach to the completed tasks. | There is evidence that most of the MIPS Assembly language tasks are complete and working appropriately.  Most of the tasks have been done with a suitably professional approach. | There is evidence that most of the MIPS Assembly language tasks are complete and working appropriately with an attempt at all tasks.  Most of the tasks have been done with a highly professional approach. | There is evidence that all of the MIPS Assembly language tasks are complete and working, although there may be issues with some aspects on one or two tasks.  Most of the tasks have been done with a highly professional approach. |
| Report | There is little or no attempt at the written discussion part of the report.  Little or no part of the report structure has been completed. | There is some attempt at the written discussion part of the report, but this part is very limited or lacks clarity and/or accuracy.  Several parts of the report structure are complete. There may be some errors in the content. | There is a reasonable attempt at the written discussion part of the report, but it is mainly descriptive and may have errors or a lack of clarity.  Most parts of the report structure are complete. There may be some errors in the content. | The discussion part of the report contains mainly description very little analysis. There may be some errors.  The report is mainly completed to an acceptable standard. | The discussion part of the report includes mainly accurate description and some analysis, making suitable connections.  The report is completed to a good standard, with use of technical language and some references, mainly presented appropriately. | The discussion part of the report includes accurate description and critical analysis.  The report is completed to a high standard, with appropriate use of technical language and of references. | The discussion part of the report includes accurate description and critical analysis showing insight in the points made.  The report is completed to a very high standard, with appropriate use of technical language and of references. |