

# Construction of an 8bit based CPU and its Boolean logic based components

# INDEX

<u>Content</u>	<u>Description</u>	<u>Page</u>
<u>Creation Process</u>		
Progression Record	OCR based document	3
Mentoring Review	A review of how mentoring was	15
Time Management	A review of how time management went	16
Presentation	The slides & review of the presentation	21
<u>Main Information</u>		
Quick Description	Overview of project in FAQ form	29
Bibliography	Reference list of sources used	30
Deciding on a title	Explaining how title was chosen	31
<u>Electronics</u>		
The Transistor	A look at the simplest components	32
Resistance	Resistance and how is it used	33
The Capacitor	Brief look and usage examples	33
Binary & Boolean	How to decode the information in binary	34
Logic Gates	Transistors joining to make logic	35
<u>General Knowledge</u>		
Background	Background information on the processor	37
Part Dissection	Using knowledge from the above section	39
FDE and Buses I	Brief look at the cycle and intro to buses	39
Registers	Look at the role registers play	40
FDE and Buses II	Combining the last gathered knowledge	44
<u>SAP-1 Construction</u>		
Clock	Controls the speed of computer	44
Bus and Registers	Implementation of buses and registers	47
ALU	The Arithmetic and Logic Unit	51
RAM	Making the memory	58
Program Counter	Where instructions are listed	62
Hex Decoder	Converting the output	64
Additional projects	Seeing what else could be done	66

## Level 3 Extended Project Qualification

*This form should be completed by the learner*

Name: Carlos Lagoa	Centre: Wymondham College
Candidate no:	Centre no:

Current programme of study: A LEVEL		
Qualification Type	Level	Subject

*Before starting this form you should have already explored various themes and ideas for your project and proposed a suitable topic to your mentor/supervisor*

Date Project started: October 2016
Project Topic: Construction of an 8bit CPU and its Boolean logic based components.
<p>Rationale: <i>why you want to do this project and how it links to progression plans/current courses/personal interest</i></p> <p><i>I want to do this project because I am eager to learn more about electronics and their past. I want to take Computer Science at university and this links nicely with the subject.</i></p> <p><i>Not only due to university, but also personally I find electronics and their logic fascinating, being able to build hardware that later on can be linked with software to make pretty much anything.</i></p>
<p>Aims and Objectives: <i>What outcome(s) do you hope to achieve. What skills are you aiming to develop?</i></p> <ol style="list-style-type: none"> <li>1. Time management</li> <li>2. Research Skills</li> <li>3. Electronics knowledge, not only theoretically but also being able to build and troubleshoot circuits</li> <li>4. Ability to express my self when doing a project that needs to inform people</li> </ol>

## Initial Planning

*What range of resources might you use/what types of research might you complete.....e.g. specialist materials/workshop space. Books/academic journals etc.*

The resources I will need will be mostly software based, I will need one or more pieces of software for the circuits creations and circuit screenshots

Research wise I will need books, most notably;

- But how do it know?, by J Clark Scott
- Digital Computer Electronics, by Albert Malvino
- Code: the hidden language of computer hardware and software, by Charles Petzold

Furthermore I will also use the internet and other software;

- Websites like SparkFun Electronics and other specialist websites, articles or journals
- Circuit simulation/rendering software and YouTube to offer more interactive solutions

What factors might you need to take into account when planning how to manage your project?

I will need to take into account the fact that there will be three main parts which can almost work independently from each other. This means that if I believe that I am stuck in a certain task I can switch to the next part and come back to that task later on. This is important to take into consideration because it will allow for more active time management which in turn could make the over all planning fail safe to an extend.

How will you plan to bring your project in for the deadline? You should create a timeline.

I will do a Gantt Chart as a way to embrace the modular approach to my project and as a way to manage my time effectively. Furthermore by having the visual aid in the Gantt Chart it will be easy to reschedule an event if needed

Supervisor

Signed

Date

## Mid Project Review

Date: February 2017

How much progress have you made in meeting your outcomes?

So far I have managed to meet all of my major deadlines, with some problems on meeting the lower priority deadlines, which appear more often than major deadlines, however it was possible to meet the major deadlines due to the Gantt Chart which allowed for rescheduling and therefore always being able to do something when the task that was supposed to be done couldn't be done.

How has your project changed or evolved?

It has changed in the way that initially I decided to design the circuits and then build them in order to prove that it worked. However, after using a variety of available software for a couple of hours I realized that it was straight forward and interactive to perform simulations, it was also simpler to troubleshoot once in a simulation software rather than in real life. With that in mind and with the intention of reducing the large work load it was decided that although some parts would be built to prove they worked the actual processor would be simulated on a software and therefore the results of the project would be in a virtual form.

Later on this idea was reinforced by the fact that I am an international student and transportation of a virtual 8bit computer is infinitely better than transportation of a physical one.

On a different note, the project has recently evolved in terms of content, with a new structure that separates everything in 3 parts.

Firstly, there is an overview of the main electronic components used in the project

Secondly, there is an overview of the history and the theory behind the processor and how it works on an abstract level

Thirdly, the first and second part come together in a series of chapters that explain how each part of the processor works

This, once more, enhances the time management by adding modular and independent sectors to the project.

What have been the strengths/weaknesses with your project management so far?

The strengths have been without doubt being able to organize everything on a modular point of view and then put it on a 'global' perspective thanks to the Gantt Chart, it has been very simple to re-organize any task given any problem arose.

Weaknesses have included the reduced electronics knowledge (experienced mostly at the very beginning of the project), selecting the appropriate software for each tasks and above all the work load in the first stages of the project as it was decided to work sequentially through the tasks instead of being able to switch between one and another, which lead to boredom (as you were always doing the same thing), that boredom then drove me to even avoiding working on the EPQ as I knew what I was going to have to do something which I was tired of doing instead of being able to move to something I found more challenging

How are you evidencing the research you have done and how you are managing your project?

To evidence the research I have done a page on the resources from which I based my self upon, and then at the end of each page I have a single line whereby I mention which, if any, resources I have relied on in order to write a paragraph, diagram, section or even page. This allows the reader to then later on go to page where all the sources are mentioned and reference it back.

I am managing my project through two main methods, the first one is the Gantt Chart, this allows me to assess my position, what should be prioritized and what can be rescheduled. Furthermore and to review my Gantt Chart and 'Monthly goals' I have meetings once a fortnight with my mentor.

With my mentor I discuss the future of the project and any arising problems I think I could have, it is beneficial to talk to him because he understands electronics and therefore can see and rank viable alternatives that I come up with to any problem, for example the problem with deciding how would I show the processor once finished if I couldn't build it all due to transportation and work load problems

What do you still need to do?

I still need to finish the main build, as time goes by I design components individually to prove their work and make screenshots of them, however - I still need to do one design whereby all the designs are joined in the software of my choice.

Apart from that I still need to do half of the last section (the one where I explain how to build certain components of the processor) and one fifth of the first section (the one where I review the electronics components)

Supervisor

Signed:

Date:

## End of Project

<b>Outcome</b>	<b>Evidence</b> - where can it be found? E.g. log book, folder etc.
<p>Exploration of a range of ideas for your project</p> <p>My project could of taken a completely different direction if it wasn't because of the research done, under the 'Information' section in the index it is possible to find a category which shows the ideas that lead to the final decision of this project and furthermore which other ideas where discarded.</p> <p>However after further researching and planning the content of a few theoretical project prototypes I soon discarded all but one of the options.</p>	<p>Such information can be found on the first pages of the write up for the project. The page that is used for this simply explains which is that I wanted to learn about vaguely, then presents a couple of ideas and explains why other ideas where not suitable and why this one was chosen. In the page it can also be found a time representation of how the ideas and project concepts evolved as research went by.</p>
<p>How you have managed your time</p> <p>My time has been managed through the Gantt Chart which later on was constructed of monthly smaller Gantt Charts which allowed me to focus on less things and therefore be more efficient. The monthly Gantt Charts where approved and coordinated (so I wouldn't fall behind either time-management wise or content wise) by my mentor, whom I visited once a fortnight.</p> <p>If we were to split our visit monthly, the first meeting would mostly be about approving my month-long Gantt chart and explaining why I would want to move something far away or closer to me (if I ever did), then the second meeting was about simply checking how I was doing and if I was managing it okay. I could also ask for help in certain areas, for example on how to explain concepts easier.</p> <p>Overall the Gantt Chart-Mentor method of time management was excellent. In my opinion, it was friendly and approachable enough for me to propose changes and ask for help, but it was</p>	<p>Within the write up and under the 'Information' section, with the title 'Time Management' it is possible to see the Major Gantt Chart, along with the monthly Gantt Charts composing it. Furthermore it is also possible to see the outcome of the meetings with my mentor as well as any changes on the original schedule.</p>

<p>also challenging and well done enough for me to not often want to do changes or ask for help.</p>	
<p>Use of a wide range of research and resources</p> <p>The evidence of me using a wide range of research and resources lies everywhere in the project.</p> <p>An example of this are the visual aids used in the project, almost every diagram has been based upon the original diagram which was found in the information researched. Please note, by saying information researched I mean chapters in books, articles online, first hand research and videos mostly.</p> <p>The way things are explained in the project makes it, hopefully, possible for someone non-specialist in the matter to understand what is going on. Therefore I have used a lot of examples and metaphors which relate to real world examples.</p> <p>Like it is commonly said, the best way to ensure you have learned something is to be able to explain it. Therefore and with that in mind I consider my project to have a wide range of resources and research as it has a wide variety of friendly explanations written by me, which I could not of written if it wasn't because of the wide research and points of view I have experienced whilst working on the project.</p> <p>Furthermore, the project has three main sections, and each section has to reach to a different branch of information gathering in order to be able to exist. For example, the second part of the project (the one that links to the past, and presents the abstract theoretical use of the components) used mostly the information from the three main books that I used in the project, whereas the third part (the one whereby, and thanks to the before-gathered knowledge, explanations on</p>	<p>The evidence for the direct use of resources can be found in the bottom of the page whereby there is a list of the sources used.</p> <p>Later on, if the reader is interested more in the resources and research material there is a specific part, on the first couple of pages, whereby it is possible to get more information on each item of research, such as books, articles and videos.</p>



<p>how each component work and is constructed are shown) used mostly the help from other sources such as videos and visual aid more than anything.</p> <p>Each of the three parts couldn't exist without the other two, however they all offer different aspects of knowledge (one more fact based, one more abstract and another one more imaginative) to the reader and that is what makes the project use such a wide variety of sources.</p>	
<p>Evaluated research and selected appropriate research</p> <p>During the time where I decided the title of the project and afterwards where I decided the content of the project I used a wide variety of research sources. This allowed me to plan well towards what my project should look like which in turn helped me with time management and furthermore knowing which of the wide research material I should keep near by in order to write about it later.</p> <p>Also, during the time where I was actually writing the chapters, and because my project is aimed to non-specialist or specialist wanting to learn more in depth, I had to do extensive researching since I wanted to base my writing upon something which was easy enough to understand for me whilst reading it that I could mold the writing to my own writing style and add my own examples and diagrams. But it also had to be a source which was interesting, challenging and useful for the future, so that the specialist could also use this as a base for his own extended knowledge.</p>	<p>A simple example of this could be that whilst doing the labeling for the circuits I had to use the specialist terminology (<math>V_{cc}</math> instead of something like 'Battery Here' for example), this was so that if there was anyone that wanted to further on their knowledge after reading my project they wouldn't find it difficult to understand how components they have already seen are labeled.</p> <p>Another example of how I have evaluated my research can be seen in the Bibliography, where information is being gathered from places ranging from hobbyist electronic websites to scientific journals. As sections came to an end I chose slightly more advanced material simply because the reader had gained knowledge as he read through the chapters on the section, thus giving me the opportunity to select more knowledge-appropriate research.</p>
<p>How you made decisions and solved problems</p> <p>Decisions, such as the one on what content to have on my projects, were made after deciding the overall aim of my project and what I wanted to achieve. After I had a clear idea on my head</p>	<p>The biggest evidence of decision making and problem solving is within the 'Mentoring' section, whereby there was a problem that had to be handled, and that was the increased work load attached to physically</p>

<p>of what it was that I wanted to achieve it was really easy to make decisions which before could seem difficult. For example the decision of knowing which sources I wanted to use was fairly straight forward and non-problematic after knowing that my aims where to inform the non-specialists and further on the knowledge from those that are specialists in the area but not on this topic.</p> <p>As for problem solving, the way problems were solved was slightly similar to how decisions were made in the way that the final aim was taken into consideration. In the case of problem solving the aim of the project and its content where presented against the problem and an interpretation of how much the final project aim was followed under the possible solutions was done, the solution that changed the final project less was the one that was followed. When making the decision other factors where taken into account such as the increased workload that the given solution could bring (and time consumed) or the repetition of content if the project was regarding content.</p> <p>The outcome of using this generalized idea for solving problems and making decisions allowed everything to be put in the same perspective (as every issue and decision was treated the same). After a couple of decisions where made and the method had been carefully followed every problem appeared not as an actual problem but as a challenging quest to carry on with the project just as we would normally do. Problems were solved faster and decision were arranged better, which resulted in an overall increase in the quality of the project in the given time we had.</p> <p>Please note, sometimes solving a problem and coming up with a decision where the same thing so both methods where used.</p>	<p>building the processor and the fact that I could only get a chance of building the processor for around 1h or 2h a week.</p> <p>Once this problem was discovered, there were a series of solutions offered; the two strongest ones where to either remove the processor from the Electronics Club (which is where I physically build my components, as there is a safe atmosphere and a large variety of electronics to use) and have it with me at all times or to build it on the a special software and have it on paper.</p> <p>The final decision was to have the processor within the software, as physically carrying a processor with me was incredibly inconvenient as I am an international student.</p>
<p>How you used technology - where appropriate</p>	<p>The evidence for this can be found along the project, especially at</p>

<p>Within the project I used technology for a variety of things, of course to actually write the project and research it a computer was used but furthermore there were a variety of software solutions used throughout the project.</p> <p>I used mainly two different programs at different stages of the project in order to represent and simulate the circuit designs that I needed to do as the project went on. Furthermore when I was building physical components to prove that what I had written worked I used one more piece of software in order to convert my designs into the appropriate circuit models, so that then I could print it and build the circuit.</p>	<p>'SAP-1 Construction' section which is filled with designs of circuits that carry on the tasks that the computer has to do.</p> <p>Sometimes the designs are more or less abstract, but that simply changes according to the thing being explained and how abstract the explanation is.</p>
<p>Presentation given to an audience</p> <p>There was a presentation given to the audience which aimed to cover two different parts.</p> <p>The first part was the project by itself, whereby I explained what the project was about and how it could personally help me in the future. The second part was about the process of creating the project, this means that I talked about what I learned from managing such a large project, what things I considered that went well and not so well and what would I change in the future if I were to do it again.</p> <p>I used the presentations for two different things; I used the 'project' part of the presentation as a way to check that everyone understood what I was trying to explain as I was trying to show them some of the electronic concepts discussed in the project. As my audience in the presentation were non-specialist mostly, if they could understand what I meant then non-specialists reading my project would too understand what I meant which was one of my initial aims.</p> <p>Secondly, I used the 'process of creation' side of the presentation in order to assess the external</p>	<p>The slides of the presentation and the reviews done by the audience are located within the 'creation process' section in the project.</p> <p>The reviews are made by teachers, however I have also received very helpful feedback from colleagues who filled in the same paper as they where in the presentation.</p>

<p>opinions about how I managed my project, my time and my resources. This assessment could be reinforced when at the end of the presentation I had the chance to be asked more questions and answer them (mostly about the process of creation).</p> <p>Furthermore I was also given a sheet of paper containing what the audience thought of my presentation, this gave me a view on how other people viewed the project and gave me a very good insight into where I was in terms of friendly language and approachable content.</p>	
<p><b>Making links to HE/Career</b></p> <p>I am hoping to study Computer Science in university and in special Robotics, as it is an extremely interesting subject and one of the fastest developing industries right now, furthermore Robotics combines the software-side of knowledge and the hardware side, which is something that not all Computer Science careers offer (web development for example).</p> <p>It is because of this that this project will help me in the nearby future as it has laid the foundation on what I have to learn regarding hardware. After this I can go on and learn more about software and hardware in separate, but the opportunity to do this project has given me the chance to do something I hadn't done until now, which is realize how they combine on the most basic level.</p>	<p>The evidence of the links I want to make with Robotics are everywhere in the project but specially in the last section, 'SAP-1 Construction' where the working of components still present in nowadays computers are explained.</p> <p>The last chapter of the last section then explains how the project could be extended and how I would want to extend it which once again shows my interest to the career I want to do and how this project links with it.</p>
<p><b>How you have reflected on your project management skills and the quality of your outcome.</b></p> <p>The project was been a pleasure to do, it is extremely accurate in regards to the initial aim planned months in advance and that just denotes how useful it was to plan in advance and use Gantt charts to be able to have a span of time to do everything, even take a break.</p>	<p>The evidence for this is on the final result, which offers a wide variety of topics and how the Gantt chart and mentor meetings reviews went on (without any major problems and with, in the case of the Gantt chart, a wide span of time to be able to reschedule the items in the monthly chart)</p>

<p>Mentoring was another great idea as it allowed me to have a perspective point of view on the overall project and therefore there was always someone that could assess my project apart from me, which was extremely useful for things like preparing the presentation.</p> <p>Through the excellent time management and mentoring my project grew in quality and furthermore dissolved any problem that came up in less than one week (which was enough for me to spend time thinking about a solution, and then working on it to later on present it in the meetings with my mentor)</p> <p>There were no major problems with either the content and time management and the project was written with a wide range of research, which only added the quality of the project, as I could focus all my time into writing the content and not into being always solving problems due to planning problems.</p>	
<p>On-going evaluation</p> <p>The process of the creating the project was very eye opening as it was the first independent project I did, I further developed my research and writing techniques and had a go for the first time at time managing which went relatively well, any problems that were not foreseen in the initial planning and then appeared throughout the project were solved to a really high standard and I'm confident I will only improve from now on.</p> <p>If I were to do the problem again I would design more independent sections and equally sized chapters, as this was one of the biggest problems I faced; allowing 4 days to write about chapter 'a' - lasting 2 pages, and then allowing the same time to write about chapter 'b' which lasted 7 pages, Overall the creation process was smooth and problem free.</p>	<p>The evidence for this can be seen in the comparison between the Gantt chart and the write up, which don't have so many differences and therefore shows the careful planning and extensive problem solving and decision making that went into the project. The evidence for this can also be found on the final evaluation.</p>
<p>Date of submission:</p>	<p>Student signature:</p>
<p>Supervisor signature:</p>	

## Final Evaluation

Overall I was extremely happy with the project and how it went. This experience has prepared me for bigger projects and how to handle them, this will be especially useful in university as there are large projects to work on. I have been able to extend and enhance my time managing skills, an example of this could be that, before when I wanted to check my progress on a project, I would look at how many pages I'd written, but now, that I have been able to develop my time managing strategies, I look at the progress on the Gantt chart or time keeping equipment I have. Furthermore, not only has this prepared me for university in terms of managing but also in terms of the actual knowledge. I now have more background on the basic electronic functioning of a computer, this means that once in university I will be able to extend this knowledge.

There are many things that went well. One of the things that went the best was the tone and user friendliness of the project in terms of wording and diagrams as it greatly reassembled the original aim of making this project for a non-specialist. This was reinforced when in the presentation I used the tone and definitions that are used in the project and people understood what I was saying. This is extremely beneficial for the project as it is essential that it is interesting and understandable. Another thing that went extremely well was the modularity of the project. This refers to how the three sections were so modular from each other that if one was to read only one section, the reader would understand what is being discussed. However if he were to read all three he would be able to appreciate how the knowledge extends from section to section.

There are of course things that I would change if I had the chance to, for example, I would make mentoring meetings happen when in need and not upon a fixed 14 day period. This is because sometimes I had a problem that I needed to discuss, and I had to wait around 10 days to be able to speak to my mentor again, this wasn't such a large problem however because I could move about the things I had to do so that I would always be occupied with something to do, however it is definitely something which I would change. Something I would change is my writing style when it came to the third section as I would start writing before I had read all the material I needed to read about, that meant that often I had to delete entire pages because I found better and more efficient ways of explaining the same concept.

Even though the project went extremely well, there are some things I which would of been even better if I would of taken a different course of action. In special I would of liked to make the decision of not making the physical build a long time before I did, this would of made the project better because I would of had more time to focus on the parts which I was actually going to present.

The presentation I did on the project went well too, the slides and reviews can be found later on, but overall I was really pleased. I was nervous so presentation skills like eye contact could of been better, however I was happy with the mark I got. The explanations were understandable and the answers to the questions asked by the audience where quick and to the point, answering their questions.

As final thoughts I would recommend to anyone taking on an electronics project to base themselves from as much information as they can, I struggled to do so for the initial stages and once I started doing I felt really confident with the work I'd done and with all the information I had gathered. I look forward to extending my knowledge and finding ways to apply it to my Computer Science course, especially in the robotics branch.

## **CREATION PROCESS: MENTORING REVIEW**

Mentoring played an important role in the process of creating this project. It helped with things such as time management and content in the project. Meetings with a mentor occurred once a fortnight, the first fortnight of the month was to create a monthly set of tasks I would have to carry on, and the second fortnight of the month (mid month) was mainly used to review what I'd done so far and solve any problems that came up, typically within the last 2 weeks. Here you can see a review of the meetings with the major occurrences being listed. The meetings happened on Wednesdays, which is when Electronics club was on, due to vacation and other reasons some meetings were done as emails.

### NOVEMBER 2 MEETING

This meeting was used mainly for talking about what I wanted to achieve with the project.

### NOVEMBER 16 MEETING

This meeting was used to review and get a better idea on the content which the project would have, once the final content was decided a few days later I started the write up.

### DECEMBER 7 & 14 MEETING

This meetings were canceled due to mocks.

### JANUARY 4 MEETING

This meeting was simply used to review how the write up was going on the first section, nothing major was changed. The 'Resistance' part of section one was shortened a bit after some talking. The best software was also reviewed for doing a variety of actions, to finally decide that more than one program should be used.

### JANUARY 18 MEETING

This meeting was about the start of write up on a new section, General Knowledge. We talked about how much history we would want that section to have and why. Furthermore I also did some hardware building on Electronics club and was assisted by my mentor when I ran into problems.

### FEBRUARY 8 MEETING

This meeting was mostly about the General Knowledge section and about polishing off anything that could be considered inaccurate. A quick run of a 'prototype presentation' which I prepared last month was done for my mentor and opinions were given.

### FEBRUARY 22 MEETING

In this meeting we started talking about the work on the last section, the SAP-1 Construction. Furthermore sections one and two were checked for mistakes and the design of some basic components started digitally.

The biggest change that this meeting brought was the decision of presenting the final build of the processor in the form of paper as opposed to presenting an actual physical build. This was decided due to workload problems, due to the fact that I wasn't able to access the physical build that often if I didn't have it with me, but if I had it with me then I wasn't able to transport it with me because I am an international

student.

#### MARCH 8 MEETING

This meeting simply focused on reviewing how the write up on the Construction side was going.

#### LAST MINUTE MARCH 14 MEETING

This small meeting was a last check on the presentation which I would have to give the following day.

#### MARCH 22 MEETING

This meeting was one of the last formal meetings that we would have, we reviewed any problems that the Constructions section brought around (mostly software wise). After this meeting we arranged to speak once I was back in school, April the 19<sup>th</sup>, where I would have finished the project and done the EPQ evaluation.

#### APRIL 19 MEETING

This meeting simply was to have a small check on the project and allow my mentor to sign the appropriate paperwork.

### **CREATION PROCESS: TIME MANAGEMENT**

This project was managed by a series of Gantt Charts, there was a major Gantt Chart, which spans for the whole duration of the project and measured milestones, such as “finished chapters on electronics”.

Furthermore there is a series of smaller Gantt Charts, intended to measure the shorter spans of time, lasting around month. This smaller Gantt Charts were reviewed in meetings once a fortnight with my mentor.

In the next couple of pages we will be able to see all the Gantt charts that were done, the diagram below is the global Gantt chart, orientated in landscape as it was originally designed, after that the monthly Gantt charts will be shown.

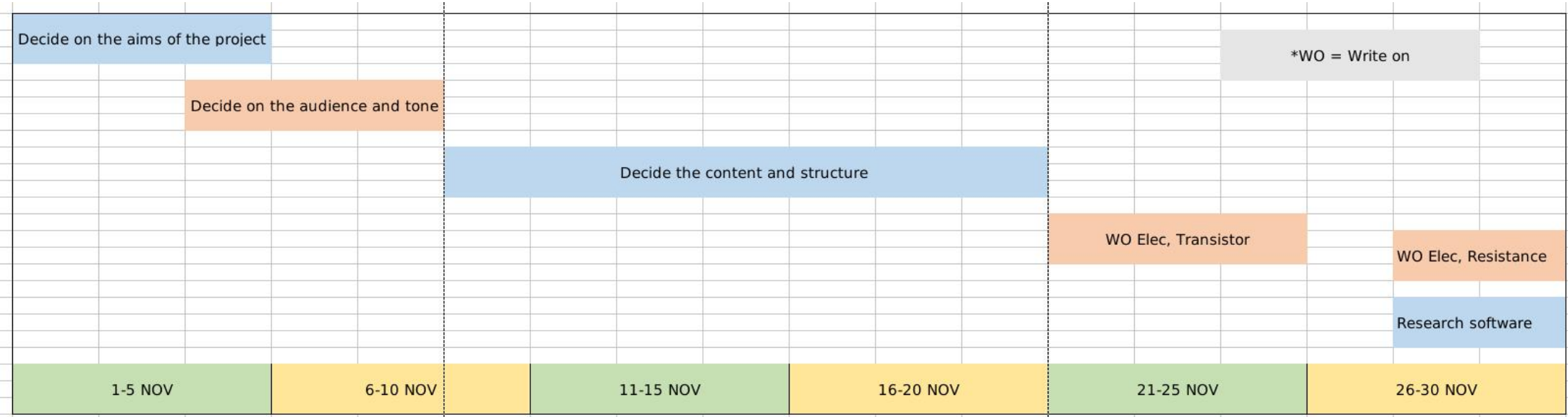


## GLOBAL GANTT CHART



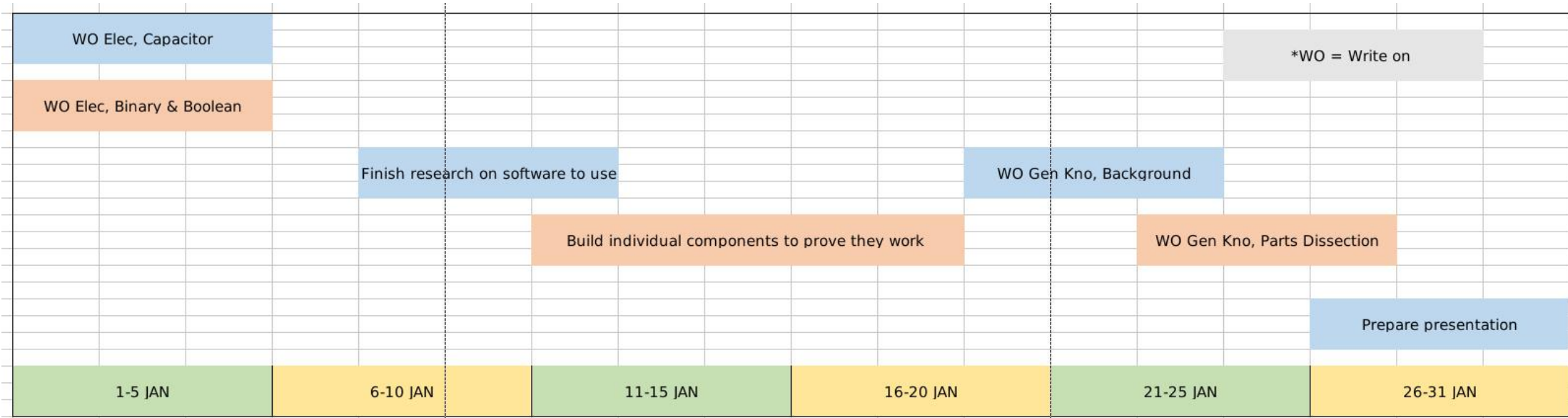
The colors displayed in the chart have the following meaning

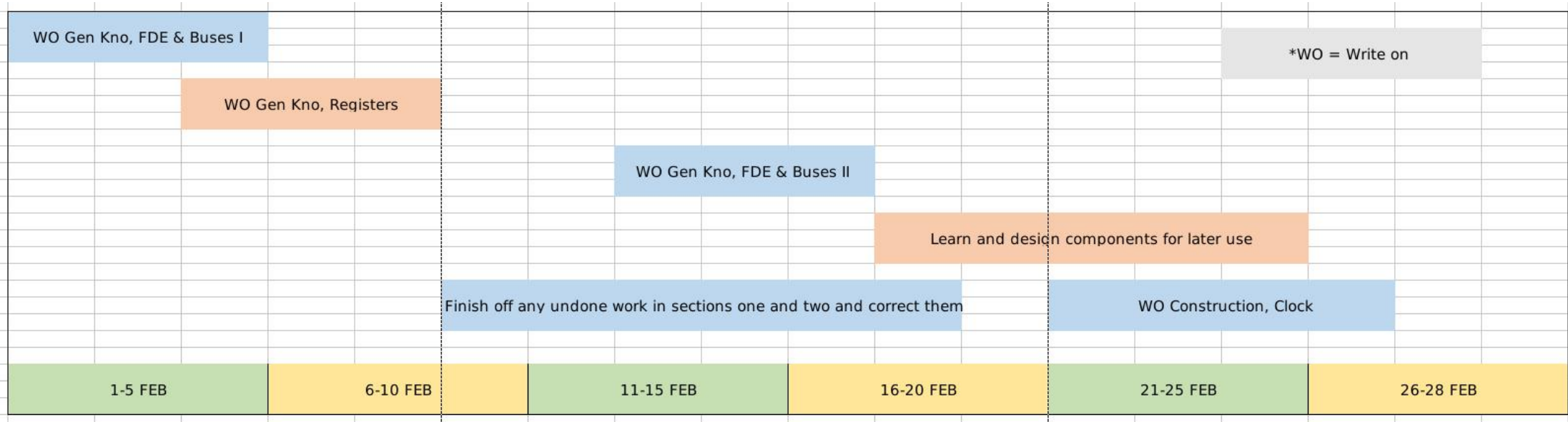
- Title Decided! Start of project (end of October)
- Decide on structure and content, aims, audience and start working on resource gathering
- Work on Electronics section, test different software for building task, start looking at presentation and start building simple hardware
- Start building individual parts and work on General Knowledge area
- Finish off any unfinished parts on sections 1 and 2 and correct them, do section on construction, finish screenshots of components, prepare presentation and finish project progression report.



**MONTHLY GANTT CHART NOVEMBER (up)**

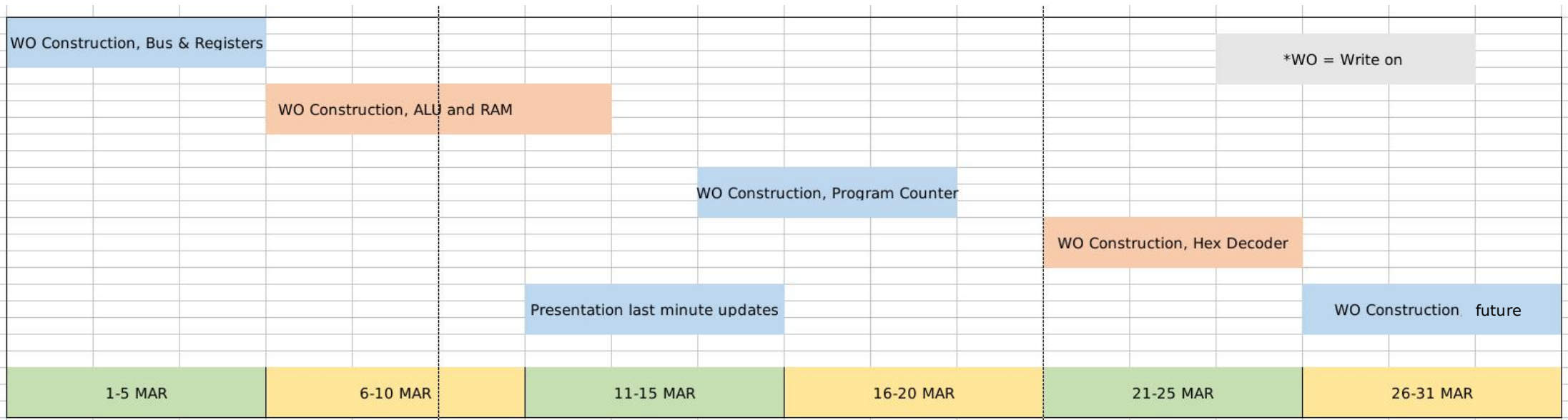
**MONTHLY GANTT CHART JANUARY (down)**





**MONTHLY GANTT CHART FEBRUARY (up)**

**MONTHLY GANTT CHART MARCH (down)**



### MONTHLY GANTT CHART APRIL



This is the last Gantt Chart made and only covers about half a month as the deadline was days after the 15<sup>th</sup> of April. As it can be seen the Gantt charts along the months are made up of little block which allow for rescheduling in case it is needed.

Overall, I was very pleased with time management as it was interesting enough for me to be engaged, but also had some breaks which allowed me to relax and not get too stressed, like for example, when there was a couple of days where I didn't have to write anything in mid January and I only had to test components. That 'free time' allowed me to sit back and get ready to do more work, when in fact I was still testing components and therefore doing work.

# CREATION PROCESS: PRESENTATION

1

## Construction of an 8bit RISC based CPU and its Boolean logic based components

\* Carlos Lagoa

2

### What am I going to talk about

The project	The process
What the project is about	What have I learned about learning
How the project excites me and interacts with my future	How did managing go?
What went well and what didn't	What skills have I developed
What might I do differently next time	Problems and how I solved them

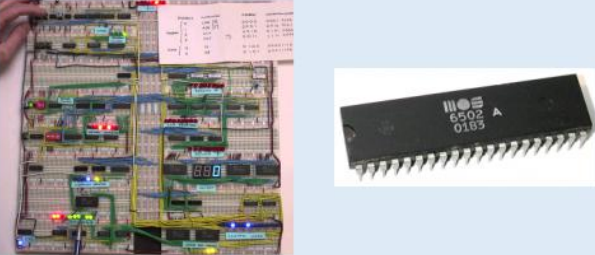
3

### What the project is about



4

### What the project is about



5

### How the project excites me and interacts with my future

- \* Electronics
- \* History of subject; David Packard & William Hewlett and Steve Jobs & Steve Wozniak
- \* Will allow me to know the basic functioning of processors and potential use
- \* University

6

### What went well and what didn't

What went well	What didn't
Varied research material	Computerized design stage
Resource gathering	Short circuits, over voltage...
Different opinions and views	Design and layout translation
Physical Construction	Electronic data sheets
	Troubleshooting
	Unscheduled errors and delays

7

### What went well and what didn't

What went well	What didn't
Varied research material	Computerized design stage
Resource gathering	Short circuits, over voltage...
Different opinions and views	Design and layout translation.
Physical Construction	Electronic data sheets
	Troubleshooting
	Unscheduled errors and delays



8


### What went well and what didn't

What went well	What didn't
Varied research material	Computerized design stage
Resource gathering	Short circuits, over voltage...
Different opinions and views	Design and layout translation.
Physical Construction	Electronic data sheets
	Troubleshooting
	Unscheduled errors and delays

9

### What went well and what didn't

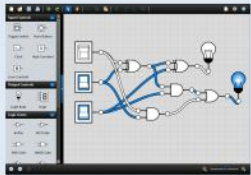
What went well	What didn't
Varied research material	Computerized design stage
Resource gathering	Short circuits, over voltage...
Different opinions and views	Design and layout translation
Physical Construction	Electronic data sheets
	Troubleshooting
	Unscheduled errors and delays





10

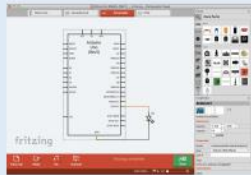
### What went well and what didn't



Computerized design stage  
 Short circuits, over voltage...  
 Design and layout translation  
 Electronic data sheets  
 Troubleshooting  
 Unscheduled errors and delays

11

### What went well and what didn't



Computerized design stage  
 Short circuits, over voltage...  
 Design and layout translation  
 Electronic data sheets  
 Troubleshooting  
 Unscheduled errors and delays

12

### What went well and what didn't

Varied research material  
 Resource gathering  
 Different opinions and views  
 Physical Construction

Computerized design stage  
 Short circuits, over voltage...  
 Design and layout translation.  
 Electronic data sheets  
 Troubleshooting  
 Unscheduled errors and delays

13

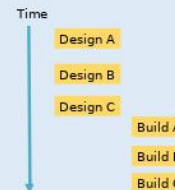
### What might I do different next time

- Design / Build Structure
- Naming and categorisation of resources
- Fix problems with ordering shortage

14

### What might I do different next time

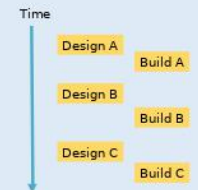
- Design / Build Structure
- Naming and categorisation of resources
- Fix problems with ordering shortage



15

### What might I do different next time

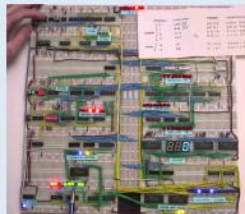
- Design / Build Structure
- Naming and categorisation of resources
- Fix problems with ordering shortage



16

### What might I do different next time

- Design / Build Structure
- Naming and categorisation of resources
- Fix problems with ordering shortage



17

### What might I do different next time

- Design / Build Structure
- Naming and categorisation of resources
- Fix problems with ordering shortage



18

### What might I do different next time

- Design / Build Structure
- Naming and categorisation of resources
- Fix problems with ordering shortage

19

### What have I learned about learning

- Value of varied up-to-date resources
- Value of planning
- Value of making notes and summarising
- Not every learning method is out there

20

### How did managing go?

- Gantt chart
- Mentoring
- Planning made it useful to distribute workload
- Problems that came up
- Combining research, knowledge, design and documentation

21

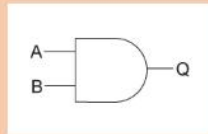
### What skills have I developed

- Electronics knowledge
- Understanding and linking of concepts
- Ability to express myself better
- Manual work on electronics
- Troubleshooting

22

### What skills have I developed

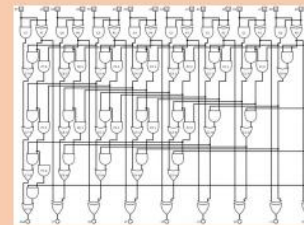
- Electronics knowledge
- Understanding and linking of concepts
- Ability to express myself better
- Manual work on electronics
- Troubleshooting



23

### What skills have I developed

- Electronics knowledge
- Understanding and linking of concepts
- Ability to express myself better
- Manual work on electronics
- Troubleshooting



24

### What skills have I developed

- Electronics knowledge
- Understanding and linking of concepts
- Ability to express myself better
- Manual work on electronics
- Troubleshooting

25

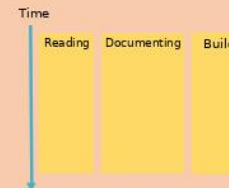
### Problems and how I solved them

- Not knowing where to start
- Software difficulties
- Documenting stage
- Time managing

26

### Problems and how I solved them

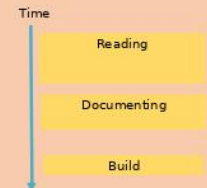
- Not knowing where to start
- Software difficulties
- Documenting stage
- Time managing



27

### Problems and how I solved them

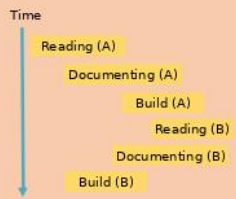
- Not knowing where to start
- Software difficulties
- Documenting stage
- Time managing



28

### Problems and how I solved them

- Not knowing where to start
- Software difficulties
- Documenting stage
- Time managing



29

### Problems and how I solved them

- Not knowing where to start
- Software difficulties
- Documenting stage
- Time managing

30

### Problems and how I solved them

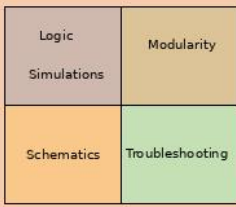
- Not knowing where to start
- Software difficulties
- Documenting stage
- Time managing



31

### Problems and how I solved them

- Not knowing where to start
- Software difficulties
- Documenting stage
- Time managing



32

### Problems and how I solved them

- Not knowing where to start
- Software difficulties
- Documenting stage
- Time managing

33

Thank you for the interest in this project;  
are there any questions?

In the following pages we can see the reviews that were made based on the presentation



placeholder 1a

placeholder 1b

placeholder 2a

placeholder 2b

## **MAIN INFORMATION: QUICK DESCRIPTION**

### **What is it about?**

This project is about understanding the components that make a very simple 8-bit computer. After understanding the components, we combine them and design the 8-bit computer, called the SAP-1

### **Why is it called the SAP-1?**

In the book Digital Computer Electronics, by Albert Malvino, he talks about a very simple architecture (that means the way components interact with each other) called the SAP-1, Simple As Possible, which is the one that we are trying to replicate

### **What does 8-bit mean?**

8-bit simply defines that the maximum size the computer can operate is a value (in denary) combined of 8 bits (in binary), this means that, for example, the maximum addition this computer can do is an addition with the result of 256.

### **How is this project structured?**

This project counts with 3 sections which vary in content but which need each other in order for what is covered in them to make sense.

The first section simply introduces the user to the most common electronic concepts, putting a meaning on terms like resistance or explaining how a transistor works.

The second section expands the reader's knowledge on another direction by introducing an abstract and theoretical layer which is used to explain the functioning of every component in the processor as well as giving some background knowledge.

The third section joins the knowledge gathered in the two sections above and simply instructs the user how to build the components that go in the processor (learned in section two) by using simple components (learned in section one), as the chapters go by the two main ways of building it are covered.

### **Who is this project made for?**

This project is made for a range of people, ideally it is made for a non-specialist who wants to learn about the functioning of a processor but it can also be for someone who is computer science or electronics orientated that wants to know more about the subject. The project is filled with metaphors and examples for the understanding of a non-specialist in the matter.

### **How did you make this project?**

This piece of work combines research, personal interest and mentoring to make, using time management strategies, a hopefully informing and interesting project.

## MAIN INFORMATION: BIBLIOGRAPHY

Albert P. Malvino and Jerald A. Brown. 1997. Digital Computer Electronics. 3<sup>rd</sup> Edition. USA. McGraw Hill. Print. Popular book among people who want to gather basic electronics knowledge, gave me abstract knowledge and in this book the computer architecture we are for section 3 doing is described.

J. Clark Scott. 2009. But How Do It Know? - The Basic Principles of Computers for Everyone. USA. John C. Scott and Oldsmar. Print. Strong abstract knowledge, served as a bridge between the sometimes more complicated knowledge of Malvino's Digital Computer Electronics and Petzold's CODE

Charles Petzold. 2000. CODE: the Hidden Language of Computer Hardware and Software. Washington, USA. Microsoft Press. Print. Book with good reputation for understanding the beginning of computation, its language, explanations and tone where one of my inspirations.

M. D. Godfrey, D. F. Hendry. 2002. The computer as Von Neumann planned it. IEEE Annals of the History of Computing (Volume: 15, Issue: 1, 1993). Journal. A very fact based approach to the architecture reviewed in section 2.

Professor David B Haviland. 2002. The Transistor - History. Nobelprize.org Novel Media. <<https://www.nobelprize.org/educational/physics/transistor/history/>>. Journal. Background and information on the transistor, the transistor leads to the biggest part of this project; logic.

Wanlass, C. L. "Static-dynamic design of flip-flop circuits." *Transactions of the I.R.E. Professional Group on Electronic Computers* PGEC-1.1 (1952): 6-18. Journal. The journals were of great use when needing to extend my knowledge more in order to explain things better, in this case for section 3.

"A History of Commodore' s 8-bit Computers." *Low End Mac*. N.p., 13 Oct. 2016. Mar. 2017. WWW page. This article reviewed some of the popular uses given to 8bit computers, This was used on the last chapter. <<http://lowendmac.com/2015/a-history-of-commodores-8-bit-computers/>>

"How to build an 8bit computer from scratch. January 2017. <<http://www.instructables.com/id/How-to-Build-an-8-Bit-Computer/>>. WWW page. This web page and forum receiving thousands of visits was my initial inspiration for the project

"SparkFun Parts Kit." *SparkFun Electronics*. SparkFun, n.d. Nov. 2017. WWW page. Used to learn about capacitors on a basic level <<https://www.sparkfun.com/tutorials/273>>

"How Capacitors Work." *HowStuffWorks*. N.p., 17 Sept. 2007. Nov 2016. WWW page. Used to learn more in depth about capacitors. <<http://electronics.howstuffworks.com/capacitor>>

"SparkFun Inventor's Kit - V3.2." *Learn at SparkFun Electronics*. SparkFun, n.d. Nov. 2016. WWW page. This was used as a global outlook at all the major components. <<https://www.sparkfun.com/products/retired/12060>>

Electrical Resistance <<http://www.rapidtables.com/electric/Resistance.htm>>. WWW page. Factual knowledge about resistance, for section 1

HowStuffWorks. "Static RAM - How RAM Works | HowStuffWorks." *RAM*. N.p., n.d. Jan. 2017. WWW page. Knowledge extension on RAM. <<http://computer.howstuffworks.com/ram2>>

"COUNTERS." *Counters | Types of Counters, Binary Ripple Counter, Ring Counter, BCD Counter, Decade counter, Up down Counter, Frequency Counter*. DAE Notes Electronics, n.d. Jan. 2017. WWW page. This article was useful when talking about electrical devices that can count up automatically. <<http://www.daenotes.com/electronics/digital-electronics/counters-types-of-counters>>

January 2017. <<https://www.youtube.com/user/eaterbc>>. Video(s). This series of videos were used as inspiration and visual aid when learning about the 8bit computer construction, in section 3

January 2017. <<https://www.youtube.com/user/madplaysHD>>. Video(s). This series of videos were used as inspiration and visual aid when learning about the 8bit computer construction, section

## **MAIN INFORMATION: DECIDING ON A TITLE**

There have been many titles that this project could have had, of course every title would mean a slightly different structure and content. As soon as I decided to do an EPQ I knew I wanted it to be on something Computer Science related.

As Year 13 started I was involved in a series of electronic and hardware related projects, for example, I had done small projects like building an ultrasonic sensor which could sense movement in a desired distance. I even started the Electronics Club along a couple of friends, which gained popularity, bringing a couple of students per year.

With electronics slowly becoming a great part of my computer science interests I decided to do my project on something electronics related, and after much thought I decided to build a calculator and do my EPQ on that. The title was "Building a calculator with materials from the 1970's", as I had been reading in special about the start of the personal computer and microchip revolution, with companies such as HP and Apple settling in Silicon Valley.

I furthered my knowledge on how a calculator worked and discovered that although extremely interesting, the idea of building a calculator had a flaw. It was either minimal work load (for the length of time given to complete the EPQ) due to the fact that simple calculators were, in fact, fairly simple to build yet very repetitive, or an extreme work load, this is because of the scientific branch of calculators which started with the HP-35 in 1972. Scientific calculators are far too complicated to talk about for the time given in the EPQ. Therefore the idea of the calculator was dismissed, yet another idea came up.

As I became more and more interested in the 1970's and 1980's computer industry, even reading the Steve Jobs biography (which talks about the personal computer industry and stories as such in that time) I decided that I wanted to build something that wasn't far off from a calculator, I wanted to build the brains of a calculator. Therefore from that point on the title became "Building a processor"

As microchips became smaller and cheaper, such as the INTEL 4004, they were put into calculators, computers and even military planes. After some research I decided to base my processor on an 8-bit MOS Technology 6502 microchip, released in the 1975, which revolutionized the personal computer industry due to its price. After that the project was given its final title, "Construction of an 8bit CPU and its Boolean logic based components".

## ELECTRONICS: THE TRANSISTOR

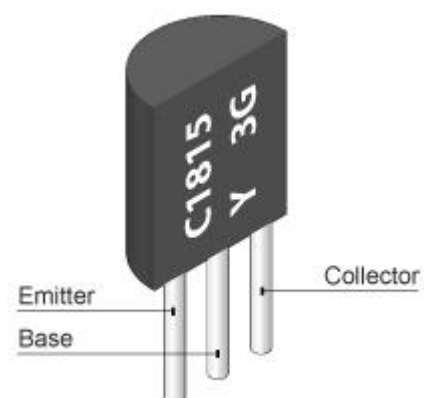
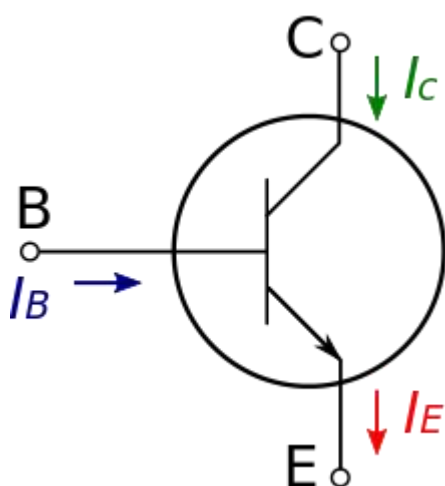
To fulfill the knowledge that this project requires, it is necessary to know about a few electronic components and concepts – one of them being the transistor.

The transistor was first invented in the late 1900's, but its electronic application only began in the late 1940's in Bell Labs by J. Bardeen, W. Shockley and W. Brattain. The transistor can be thought of as switch but with a few new properties; firstly, it requires no moving parts, secondly, it can be functional on a microscopic level and third, it doesn't need interaction from humans to work (for example someone turning it on and off)

Transistors have three parts, the 'Receiver' can be thought of as the input, then the 'Emitter' can be thought of as the output, for last the 'Base' which is a part that makes a transistor give a signal or not. Transistors play an important part in electronics as it is a cheap and effective way of performing two tasks; switching and amplifying.

Firstly it can act as a switch, this is thanks to the fact that the base of the transistor is a semi conductive material and therefore provides a path for the electrons at the receiver to travel to the emitter. Please allow a bit of abstraction to understand the concept; a current of  $x$  can be applied to the base so that a current of  $150x$  can flow from the receiver to the emitter.

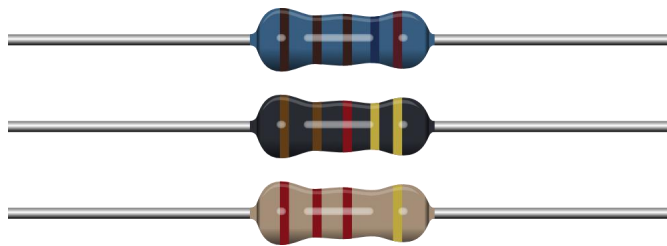
Secondly, a transistor can act as an amplifier, this is when a signal of  $x$  is received at the receiver and a signal of  $2x$  is given off at the emitter. An example of a practical use of this can be with a 'hearing aid', where a signal of  $x$  'noise' is received into the machine and then a signal of  $2x$  'noise' is played through the speakers for the person wearing them. More on how this effect actually takes place will be covered as the project follows.





## ELECTRONICS: RESISTANCE

Resistance is something which will not be covered to a great extent in this project however it is good to understand what it can do in. Resistance, in simple terms, is the discouragement of movement of electrons within the conducting material (the wire for example). By having resistance the flow of electrons is reduced and therefore the current reaching a terminal is lowered. Resistors can generate resistance on different amounts and they are widely used in circuits, for example, everytime we connect a LED as the output of any circuit (for example a button being pressed and a light bulb going on) we also put a resistor between the bulb and anything else in order to prevent it from bursting because it had too much current going through it. Here we can see different types of resistors, which have different resistances, measured in Ohms.



## ELECTRONICS: THE CAPACITOR

The capacitor is another example of an ingenious use of physics and its laws to provide a circuit with a capability that wouldn't normally be thought of. The transistor is an electronic component which allows for charge to be stored within it. This is done by providing the capacitor with two metal plates separated by a dielectric material. The two plates accumulate the charge when connected to a power source.

To discover how much charge a capacitor can store we have to look at its Capacitance (measured in Farads, F) - the capacitance is always looked at in terms of 1 Volt.

The capacitor can store a charge for a small period of time and deliver it whenever the user wants and therefore it can be very useful, an example of this is to use a large capacitor as a last case scenario emergency battery in a cold environment, where a normal battery wouldn't operate efficiently (due to cold weather and the chemicals involved in batteries). Furthermore another use could be using the capacitor to store the charge up to a point to then release its charge, informing the user that his main batteries will not work anymore due to cold weather.

Imagine if you were trying to make a light blink, by charging and discharging one or even more than one capacitor you could have a potentially very efficient light blinking automated circuit, whereby a

capacitors charges up and then delivers it charge to a switch which is connected to a 'turn the light on' button. After that the capacitors charges again and this time delivers its charge to 'tun the light off' button.

A more enhanced version of this initial concept will be used later when designing the memory for the computer.

### **ELECTRONICS: BINARY AND BOOLEAN**

Binary and Boolean are the means of communication between computational systems and basic electronic equipment, they are always two possible options to communicate, either on or off, which from now on will be referred to as 1 or 0.

Computers are able to give results and hold memory and even make decisions based on how the 1's and 0's interact with each other, this will be covered shortly. However for the time being it would be important to know how to read binary or at least how to understand what it is.

The easiest way to read binary is to think of it as a span of x numbers that are being powered from the number 2. For example the number 01011010, if beneath it we put the base 10 number span to power of the 2...

<b>0</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>0</b>
<b>2<sup>7</sup></b>	<b>2<sup>6</sup></b>	<b>2<sup>5</sup></b>	<b>2<sup>4</sup></b>	<b>2<sup>3</sup></b>	<b>2<sup>2</sup></b>	<b>2<sup>1</sup></b>	<b>2<sup>0</sup></b>

... And then we only add the 'active' bits (that is the ones that contain 1's) then we get...

$$2^1 + 2^3 + 2^4 + 2^6 = 90$$

Therefore as we can see the binary number 01011010 equals to a 90 in base ten system. Please keep in mind that more binary will be covered as this project goes along, such as decimal points in binary.

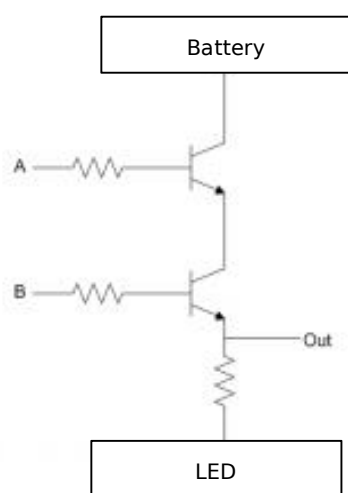
However it is easy to see how being able to convert an 'on' or 'off' into a variety of different outputs opens a completely new door to how computers operate. The concept of binary, though it doesn't make much

sense now, will come together in the following chapters where logic gates and memory will be covered in more depth.

### ELECTRONICS: LOGIC GATES

Logic gates are the most useful electronic component as they can provide logic to a circuit, this is done using a series of interconnected transistors which then provide an output based on the given input.

Logic gates use something called Boolean logic which allows it to give different output based on the different inputs. There are several types of logic gate; **AND, OR, NAND, NOR, XOR, XNOR** and **NOT**. Different gates are used based on what you want your output to be based on the input you have.



The gates are constructed by combining different layouts of transistors and based on the arrangement we can get a different output. For example to build a dual input AND gate (input A and input B) we need two transistors in series and our desired output (labeled Out) at the end of this series. LEDs tend to be the output for any circuit during its two most critical stages, development and troubleshooting, and therefore we could picture the two transistors in series along with a battery, an LED and a grounding for the circuit.

To see what would happen with the output depending on how the gates are made it is considered good practice to sketch a truth table. A truth table is simply a list of all the possible inputs and consequently its output. We can now see a couple of examples of Truth tables. In the Truth tables we write the inputs as letters of the alphabet and the result is generally labeled as 'X' - this simplifies having to write down something along the lines of 'Transistor A', 'Transistor B' and 'Result of both'.

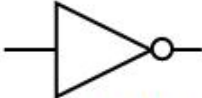
Here we have a double input AND gate Truth table.

A	B	X
A has to be on <b>AND</b> B has to be on		For X to be on
0	0	0
0	1	0
1	1	0
1	1	1

And here we can see a triple input OR gate

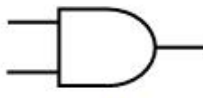
A	B	C	X
A has to be on <b>OR</b> B has to be on <b>OR</b> C has to be on <b>OR</b> all the above			For X to be on
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Here we can see the truth table and symbol for the gates that we have available.




**NOT**

Input	Output
I	F
0	1
1	0




**AND**

Inputs		Output
A	B	F
0	0	0
1	0	0
0	1	0
1	1	1




**NAND**

Inputs		Output
A	B	F
0	0	1
1	0	1
0	1	1
1	1	0



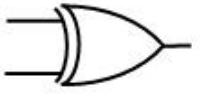
**OR**

Inputs		Output
A	B	F
0	0	0
1	0	1
0	1	1
1	1	1



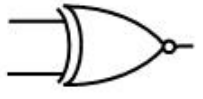
**NOR**

Inputs		Output
A	B	F
0	0	1
1	0	0
0	1	0
1	1	0



**EXCLUSIVE OR**

Inputs		Output
A	B	F
0	0	0
0	1	1
1	0	1
1	1	0



**EXCLUSIVE NOR**

Inputs		Output
A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

## GENERAL KNOWLEDGE: BACKGROUND

Jon Von Neumann was a computer scientist that is widely known for creating the 'Von Neumann Architecture'. In 1945 he wrote a paper talking about a theoretical electronical device that featured a processing unit that contained an arithmetic unit, registers and a control unit orchestrating the exchange of data between the processor and the memory through buses.

Buses can be thought of as channels which carry data between different parts of the processor or between the processor or memory. When we say between different parts of the processor we are talking about different registers.

The main difference that Von Neumann brought was the fact that one of the registers; called the Program Counter (PC), which is in charge of storing and going through the instructions one by one, not only stored instructions but actual data as well. If we imagine a bunch of lines of code, this lines of code and the data they would of interacted with would have been stored in different locations in any other architecture (such as the established Harvard Architecture) however in the Von Neumann Architecture this is not the case.

Von Neumann stored the instructions and the data they interacted within the same location. That made the construction of the computer much more easy to understand and therefore made computing more reachable to the outside world. Now we can see an example of Von Neumann's Architecture.

<i>type</i>	<i>memory</i>	<i>line of code</i>	<i>what it does</i>
Instruction	1	load mem_loc 5	instruction to load memory location 5
instruction	2	add mem_loc 6	instruction to add memory location 6 to 5
instruction	3	save in mem_loc 7	save result of operation in location 7
instruction	4	output mem_loc 8	display the result in location 8
data	5	50	this is the value of mem_loc 5
data	6	12	this is the value being added in line 2
data	7	62	this is the result of adding loc 5 and 6
data	8	Dell XPS Screen	this is where the data should be displayed

As you can see it is easier to follow now what Von Neumann meant - in easy terms, the Von Neumann architecture revolutionized computing simply because he decided to join the two most important interactive concepts; instructions and data - this simplified computing as it meant that only one bus of each type was needed (as everything was stored in the same place).

However it is all not good news – the Von Neumann architecture had a flaw. We have above mentioned the term bus and described it as a data / instruction carrying channel. Like every channel however there is only one direction of flow and therefore only one thing can travel along the buses at a given period. This can cause a Von Neumann Bottleneck, which again in simplified terms is simply when either data can travel from the memory to the processor ( or vice versa) or when instructions can travel along the processor or towards the memory and not both at the same time. If you wanted to send the number 5 from the memory to the Arithmetic Unit (just as we did with the number 20 in our above example) you would have for the bus to be free from carrying anything that could have been sent before such as the message which informed the memory that the number 5 was needed.

Although the above concept seems difficult to understand it will all make sense once we explain how the problem is solved. Lets once again recap and take a look at the problem; the problem is that since there is only one bus of each (of each, because there are 3 different buses; explained later) we can only move data in the bus one way and with one purpose; which can slow down the system if there are other things waiting to be sent down the same bus.

The solution therefore is to have some sort of method to control, or better, optimize the way the buses are used by the system. This can be easily visualized with two scenarios;

#### *Scenario 1*

There is a classroom with students; they are asked to all say shout a number whilst someone writes the numbers being shouted in order. The students start shouting with no control and the person in charge or writing is overwhelmed and can only write a couple of numbers.

#### *Scenario 2*

The same is asked to the students, but this time its asked by a teacher; which tells them that he will ask in order and then tell the person in the board which number the student said, the person in the board will then listen to the teacher and write the numbers that the teacher says.

As you can imagine it is much better for the person writing the data in the board to have some sort of control when it comes to receiving feedback from the different members of the class. This links nicely to our problem with sending information through the buses, obviously the whiteboard writer (bus) can only 'listen' to one student (CPU components and registers) in order to write the data in the board (move information from the students mouth to the whiteboard, or from register x to a certain location in the memory), therefore and in accordance to the ease of communication experienced in Scenario 2 the computer counts with a Control Unit.

The Control Unit simply put reduces the number of Von Neumann bottlenecks by a massive amount (please note there are still Von Neumann bottlenecks).

Therefore lets take a look at our problem; data to the bus appears from everywhere – and our solution; the control unit makes sure that the data received is manageable for the buses. It is especially useful in a concept that will be introduced shortly which is an example of how the control bus is of extreme usefulness.

The following concept is not a component but rather a guideline for the components to work by. It is called the Fetch-Decode-Execute Cycle and it manages the main 3 different types of interaction the different components of the CPU and the memory have to handle. But before we start learning about the FDE lets have a recap of this topic.

Whilst other architectures, like the Harvard architecture, separated data and instructions into different parts of the processor, the Von Neumann Architecture joined them, and treated them the same in the eyes of memory location. The result of this meant that a single program counter, a single Control Unit and single buses could exist within the computer, this lead to more efficient yet simpler computer designs which made it easier for people to understand computers.

## **GENERAL KNOWLEDGE: PARTS DISSECTION**

The Von Neumann Architecture counts with three main parts : -

The CPU which in more depth counts with; the Control Unit (in charge of orchestrating the exchange of data through buses), the registers (which will be covered later on, but for now are used to store very specific data which helps the FDE and Control Unit operate) and the Arithmetic Unit (which is in charge of doing any maths (such as  $20 + 30 = 50$ ) or, also covered later, any logic operations).

Outside the CPU and moving to the second main part we have the memory - which is the place where all the instructions and data are stored. This doesn't change if the computer is turned off - they will still be there when the computer is turned back on. Throughout the pages we will see about the different types of memory there are; non-volatile or volatile, RAM, hard drive (and their different types) or cache, read or read and write.

The third main concept is the buses - the buses, when instructed to do so by the Control Unit, send information required for a line of code to be processed and therefore allow the program counter (one of the many computer registers) to move to the next line (from 'load 20' to 'add 30' to the loaded value of 20')

## **GENERAL KNOWLEDGE: FDE, BUSES AND REGISTERS I**

After this recap we can now start talking about the last concept that needs grasping, the FDE - each of the letters in that acronym accounts for a very specific task that the computer does. This three tasks are done with one objective in mind; jump to the next program counter instruction so that the computer can do it all over again - thus called FDE **cycle** - this cycle is done over and over again by a computer until

the program receives a halt option, when a program counter receives a halt instruction the Control unit mandates all the components to do no more work.

Although there is a wide number of instructions that the PC can receive (add, load, halt, store; are some we have seen so far) they can all be reduced to three parts, thus the FDE cycle.

The first instruction is 'Fetch' - in this part of the cycle all the data that could be required for the process to take place is 'recollected' from memory and taken to the appropriate registers (very purpose specific data storage entities in the CPU) - lets take as an example line 1 of the example; 'load mem\_loc 5' - in this example and when run through the Fetch part of the cycle the Control Unit ensures the buses are carrying the appropriate information to the parts of the computer. We mentioned that there were 3 buses - and now it is time to go more specific - the three buses are the address bus, the control bus and the data bus - each have very specific jobs.

The Address bus is in charge of carrying the memory location of the data (mem\_loc 5)

The Control Bus is in charge of sending the 'orders; that the Control Unit sends and

The Data Bus is in charge of simply transferring information, for example the information that was read when the Control Unit informed the Control Bus to send the Address Bus to mem\_loc 5 - therefore broadly speaking we could say that the Data Bus simply carries '20'

After the buses have been appropriately used and the registers have been loaded with the information provided by the teamwork of the buses the second part of the cycle starts

The second instruction is to 'Decode' - in this part all the data is collected and therefore the computer must 'understand' what can be done with that data - a transition between the programmers representation of the data to the microprocessor's representation if you will.

The Opcode (the code in binary used to program the the function of the processor) is loaded from memory and then sent to the Control Unit. The Decode function itself is done by the Control Unit, as this component will later on 'tell' the other components what their duties are.

Lastly, there is the 'Execute' instruction, please note, when this instruction is finished the whole process starts again. However before we start on the last part of the FDE it is important to have another mid-text update on the functions of the processor, in this case the Registers

## **GENERAL KNOWLEDGE: REGISTERS**

From the above Execute paragraph you might be wondering how does the computer know where to go next, because if it didn't the FDE cycle would always repeat it self over and over again. This is where the registers start to come in to place, as we said before the registers hold very specific data for the processor to use. Along with that they also store data that has a use for only one cycle so every cycle that passes the registers are updated to hold important data for that single process.



Here we can start seeing the ‘advanced simplicity’ of computers, the importance given to a series of 3 steps and how its only three steps that are being repeated an average of 10 digits a second to create what was previously thought of as something of immense complexity. To answer our question on how do computers know which line of code to do next we can introduce the first register that we will talk about. The ‘Program Counter’ or ‘PC’ for friends. The Program Counter is in charge of keeping the computer (mix of CPU, buses and memory) updated with the current instruction it is operating, so if we were to return to our above example of instructions and absorb a PC into that model it would look something like this.

<i>type</i>	<i>memory</i>	<i>line of code</i>	<i>PC value</i>
Instruction	1	load mem_loc 5	1
instruction	2	add mem_loc 6	2

The PC references the memory location of the current instruction and therefore if we had other data in the memory and we had to store our program somewhere which wasn't the first positions of the memory the PC would display the appropriate memory locations.

<i>type</i>	<i>memory</i>	<i>line of code</i>	<i>PC value</i>
Instruction	3	load mem_loc 5	3
instruction	17	add mem_loc 6	17

It is possible to have a single program split up within the memory although this is almost always managed by an operating system so we won't talk about that.

Let's imagine that our computer has just done the first instruction and therefore fetching, decoding and executing has been done, when the computer jumps the next instruction and starts a new FDE cycle with the second instruction the PC value is updated from 3 to 17. You might wonder how the PC gets updated; it is fairly simple - the new PC value is sent by the Data Bus to the PC from the memory, so that everytime any part of the computer wants to check the PC it doesn't have to occupy the Data Bus (as its only one way) it can simply check within the CPU's register part and look for the PC.

Now that we know what the PC does it is time we look to another register. In this case we will look at the ‘Memory Address Register’, MAR.

The MAR is the register that stores one of two possible items, depending on the current part of the FDE and the Control Unit requirements. It can either hold the memory address from data that will be fetched to the CPU or the memory location to which data will be sent and stored. The MAR could therefore for example hold the value 5 in the following example:

<i>type</i>	<i>memory</i>	<i>line of code</i>	<i>explanation</i>
Instruction	1	load mem_loc 5	holds memory address from data that will be fetched

Or perhaps the number 7 in the following example:

<i>type</i>	<i>memory</i>	<i>line of code</i>	<i>explanation</i>
instruction	3	save in mem_loc 7	holds mem_loc to which data will be sent and stored

Of course the other side of the coin of the address of the data is the value of the data itself. The content holding responsibilities of that part fall into the third register which we will talk about, which can be thought of as MAR's distant cousin. It is called the Memory Data Register, MDR. This register holds the values of the memory location that will be fetched to the CPU or that would be sent to a certain memory location to be stored. Thinking as the MAR and MDR as two similar registers could look something like the following example:

<i>line of code</i>	<i>MAR value</i>	<i>MDR value</i>
load mem_loc 5	5	50
save in mem_loc 7	7	62

It is time for a small review of what we have been talking about in order to get everything in context.

The CPU is, apart from two other things that will be discussed later, formed by registers. Registers are specific memory devices that reset them self and get new values loaded after each iteration of the FDE cycle is complete. So far we have seen 3 out of 5 registers, we have seen the Program Counter (PC) - which holds the memory location of the current instruction and furthermore we have looked and 'mentally' combined the Memory Data Register (MDR) and the Memory Address Register (MAR), MAR holds the address of data that the processor or memory are or will be interacting with, and MDR holds the actual values of data that the processor or memory are or will be interacting with.

Now that we know what these 3 registers do its time to look at the last two registers - the Current Instruction Register, CIR is one of them. It is the register which is in charge of holding the instruction that has been fetched from memory. It is in fact the CIR that interacts with the Control Unit to decode an instruction.

As the last part of this section we also have the Accumulator or ACC. To talk about the ACC we have to introduce another component to the CPU, the last component that we haven't talked about and one that will surely make sense. To put things into context lets look at this line below:-

<i>type</i>	<i>memory</i>	<i>line of code</i>
Instruction	3	load mem_loc 5
instruction	17	add mem_loc 6

How does the CPU add mem\_loc 5 to mem\_loc 6, or where does it even load mem\_loc 5 to so that mem\_loc 6 can later on be added? This is where the Arithmetic and Logic Unit, ALU for the close friends comes in play - the ALU is in charge of doing calculations (add, subtract...) and also handle logic such as if statements.

An if statement, also called conditional statement is the way to add logic to programs. Let's have a very bare-bone abstract example:

age = 23

If, the age of someone is above 18...

they can drive

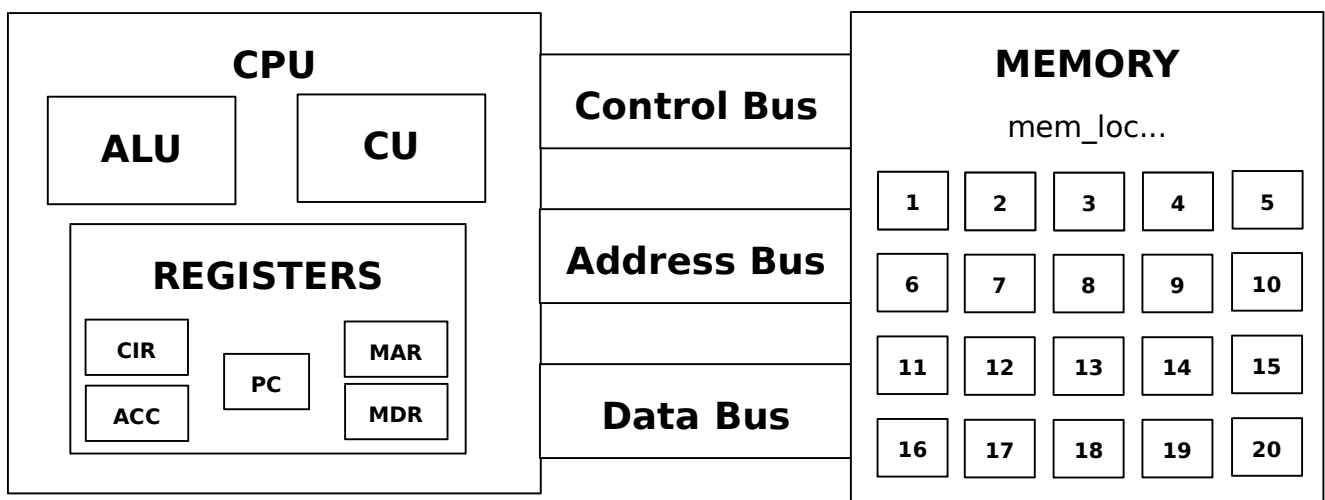
else, the age of someone is not above 18 (below 18)...

they can't drive

As its possible to see there is a certain level of logic that a computer can read, the ALU is in charge of that.

The same way we mentally joined the registers MDR and MAR, it wouldn't be a bad idea to mentally join the ACC and the ALU. The ALU is in charge of doing the calculations and logic and the ACC is in charge of two things, the first one is feeding values to the ALU, for example in the 'load mem\_loc 5' instruction the ACC holds the value of mem\_loc 5 and the ALU fetches it from there to then 'add mem\_loc 6'. The other function the ACC has is to simply store the results of calculations made by the ALU, so for example when the addition of 50 and 12 was completed, and for the duration of time between the computer coming up with the result and the user telling the computer to store it in mem\_loc 7, the result was stored and in the ACC

And that is it; those are the registers and components that make up a processor. Now that we have a little bit more knowledge we can look at the last step of the FDE. But before an abstract diagram of the parts that make up the computer so far could come in handy for getting all the ideas in concepts linked.



## GENERAL KNOWLEDGE: FDE AND BUSES II

The last part of the FDE we have to look at is the execute part. The execute part of the FDE cycle is the part where, in terms of what users expect, we see an output. The output is either a visual output, i.e. The computer displays an output in the screen after the computer calculated the result of  $5 + 4$ . Or what we could call iterative; that is an output given however it is used for the next iteration and the actual output (the visual output) will be given later on, i.e. A computer calculates the Fibonacci sequence for 10 iterations and the computer has just given an output of 3 (the 4<sup>th</sup> iteration).

When the Execute part of the FDE happens the Program Counter shifts up one value, so that the next FDE cycle that starts can Fetch, Decode and Execute the next instruction. Depending on the instruction different parts of the CPU, buses and memory. For example, perhaps the instruction is to

```
save result of 5 + 4 in mem_loc 10
```

In that case for example the MDR and the MAR would be used to store memory location and corresponding data in the memory, rather than holding the data and memory address for the CPU to load from memory. Of course it goes without saying that in the following example the jobs are switched.

```
load mem_loc 13 to ACC # this is loaded so that later on it can be added with something else perhaps
```

## SAP-1 CONSTRUCTION: CLOCK

The Clock in a computer is a critically important component, it offers the computer a rate at which to work with. Imagine the following scenarios:

### *Scenario 1*

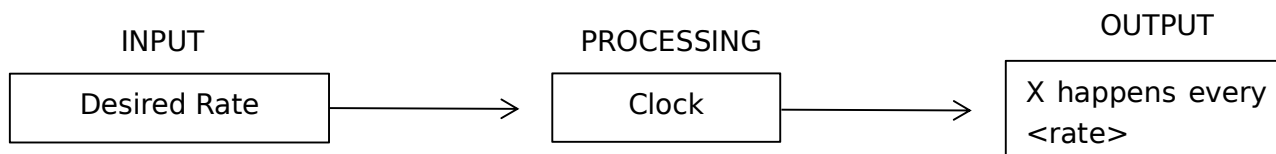
In the same classroom as a couple of pages ago we encounter the students, leaderless, shouting numbers at the board. Not only the information is too rapid to understand for the person writing, but that information which might have a chance of being heard is spoken over by another number from another part of the room and then the whiteboard writer can't seem to understand anything.

### *Scenario 2*

Now that the classroom has someone orchestrating the number output (a Control Unit) we might want to even make the whiteboard writing more efficient and, along with putting people in an order so that instructions can be heard, also tell them when to speak. This is very convenient because if the person writing is having a hard time writing on the board the rate can be slowed, and if they can handle more information then the rate can be sped up.

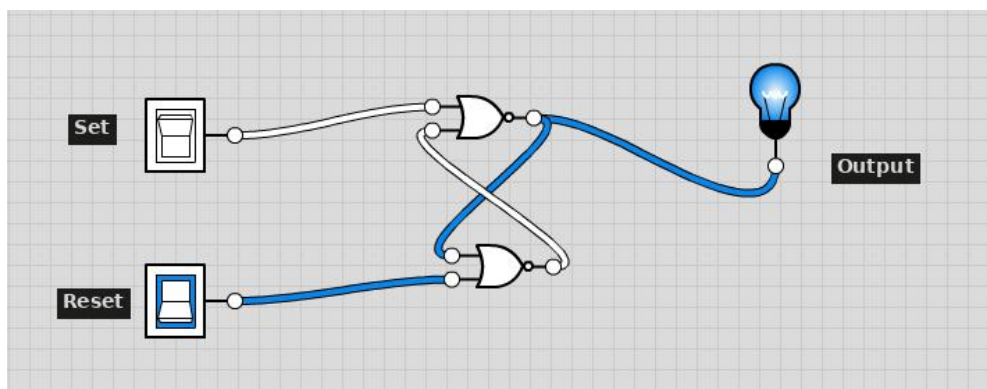
This is the case with our Clock component, we will be able to decide the rate at which we want actions to take place. An example of this could be to speed up the FDE cycle or even to troubleshoot something as we will be able to take the next step that the processor would normally take at our own pace and therefore be able to get a 'step by step' idea on how the processor does it, and see which part(s) in our code or hardware could be wrong.

To build a clock we need to get an idea of what it is that we want to accomplish. We want to be able to change the rate at which the processes happen. Therefore a concept of how this could be accomplished could resemble the following:



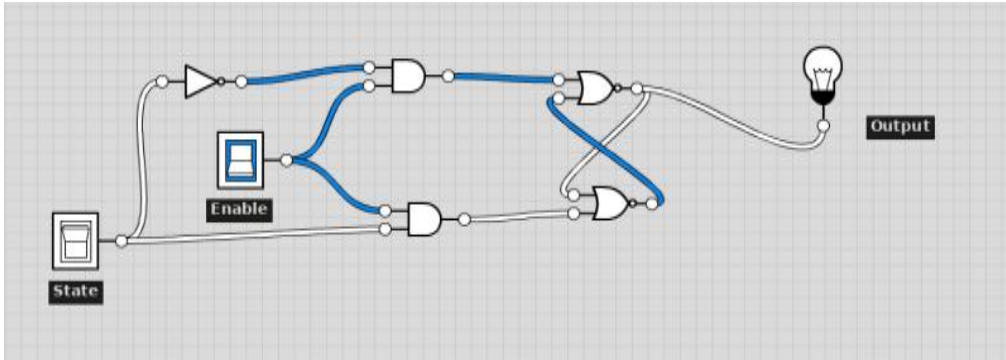
The X signifies any action that we want to link to the output, it could be an LED light, a buzzer in a fire alarm or, in our case, a step on the FDE.

To build a Clock first we have to come up with an abstract design of what we think could work. In electronics a series of components can construct something called an SR Latch, this Latch simply Sets and Resets a circuit based on the input given by using the other gate's output as one of the inputs. We will use NOR gates and the circuit then can have a manually ON / OFF state.



Here we can see that what is going on. Whilst the reset button is active we can set the output on or off, however once the reset button is disabled we can no longer change the output. This initial model is not only useful for having something set the light switch on or off but also for later on reference, when talking about keeping a state high or low (memory).

How could this circuit be more useful? It could be more useful if we only had one button to change the state and we renamed the reset to enable, when the enable is active it will be able to gather input from the state button and output it into the output, each press being an on or off. This could be seen as an extension of the SR Latch



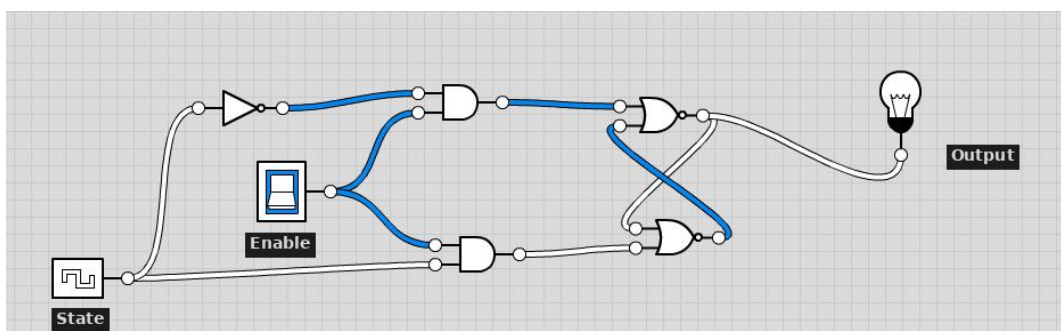
The above IC (integrated circuit is a D Flip Flop) and as it can be seen from the example (which makes more sense 'clock logic' wise) there is an enable switch, which can halt or start/resume a process and then there is a state switch, which acts as a ON/OFF switch for the output. This model although efficient presents a problem; the timing (ON/OFF switching) has to be done manually. The only way to fix this is to think of way to make the State switch change by itself after a certain time.

We are in luck however, as there is a component which will allow us to do that exactly. The capacitor allows for this to happen, along with another component called a comparator. This is an abstract of how they will work together. I will describe the step, followed by the solution and then the next step.

Firstly, imagine a capacitor charging up to a certain charge; when the charge is reached a light bulb goes on. This raises the first question, how do we know when a charge is gained? This is where the comparator comes in handy. A comparator allows for two charges to be compared - when charge  $A < B$  the output is 0, however when charge  $A > B$  the output is 1.

Now, imagine that there are two comparators and two capacitors. One holds the opposing effect of the other one; that is, one lights the bulb when the charge is above  $x$  and the other one turns off the switch when the charge is below  $y$ . That raises the second question, how would they interact? If we go back to the SR Latch example we can start to see how this would work. If we have one setting the output on and the other one resetting the output then we will be able to get a stable ON/OFF feed

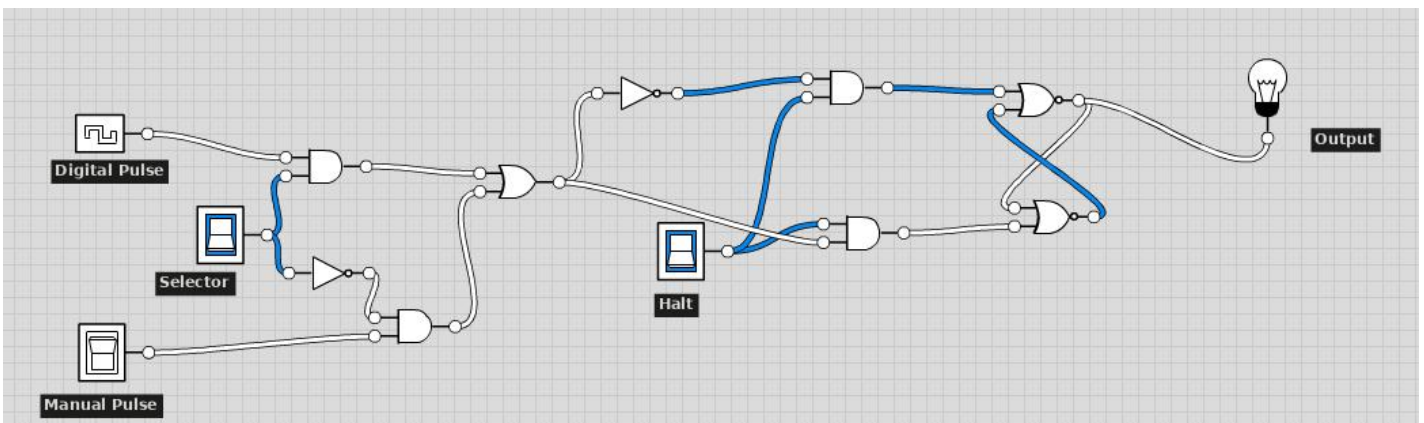
The last question that raises is how can we get a stable ON/OFF? That is when we can take advantage of the capacitors; if we can charge one capacitor, release its output over  $x$  time only to 'switch around' and charge the other capacitor, to release exactly the other output we can get a stable ON/OFF feed. This is called a Clock and we can see it in action here.



We can see from the above example that we have changed the State switch for a Clock; which is simply an automated implementation of an SR Latch using capacitors and comparators for the charging. A good knowledge link at this point could be to realize that if we change the overall resistance of the circuit the capacitors would charge at a different rate. Giving the clock the possibility to go faster or slower.

We can now get an idea of how our computer will go through the steps in the FDE cycle, if we replace the LED output with the actual computer and its component every action in the computer will be guided based upon the Clock rate.

For troubleshooting moments specially a manual pulse option has been also added, with this addition you will be able to get the computer to run at a Clock rate or to a manual pulse. We will do this the following way; adding a selector between one option or another one.



And that's it. Our clock is finished. Now we can combine it with the other components and configure at how many Hertz our clock runs, being able to also manually step through the clock.

### SAP-1 CONSTRUCTION: BUS AND REGISTERS

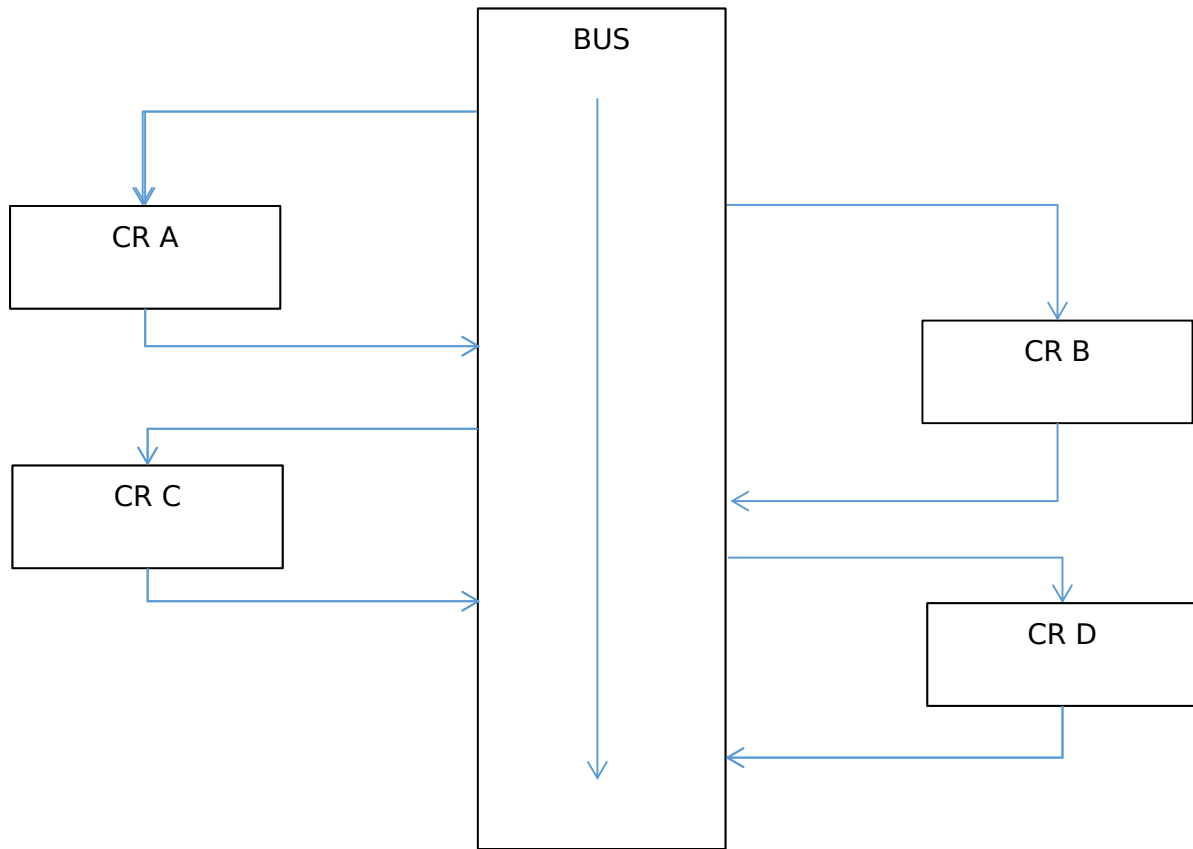
As we have previously seen there are certain types of registers that hold specific values, for example the MAR or MDR. In the case of the SAP-1, and with the aim of simplicity as one of the top objectives we will have a slight different approach to registers. We will link registers like MAR and MDR into one single register and call it a Common Register - the MAR and MDR wont be linked literally; but rather the Common Register (CR) will be able to be used as both depending on the necessity. This will be shown later on.

As we know the buses carry information from component to component and depending on the data carried on that clock cycle a register might want to 'receive' the data and another might want to let the data untouched as its not needed.

The layout we are going to use will need a certain features:

- We need a bus that will connect all the registers with each other
- We need the ability for buses to deliver and save data

To start off we will do the following abstract design:



What we see in this diagram is a very simple abstract model of the bus-register interaction that we need. Every CR has access to any other CR through the bus, now imagine connecting this to our already built clock and we have the bus 'checking' if any register has something to 'say' every clock rate. This amplifies the previously aforementioned point whereby we slowly realized that the computer just does really 'simple' operations at a really high speed (for example a 2.5 GHz processor does 2.5 billion of this clock cycles per second)

The bus we will be building will be 8-bit; this means that the bus will simply be a recollection of 8 cables that split into every register sequentially. The registers will therefore also be 8-bit.

Even though we don't have an actual logic diagram we can see how to transfer an 8-bit value (8 cables holding a single bit each) from CR A to CR B. The 'transfer cables' leave the bus structure to go through CR A, once they get to CR A they cables state gets set to whatever CR A is (01011010 for example). Then the transfer cables rejoin the bus and travel down to CR B.

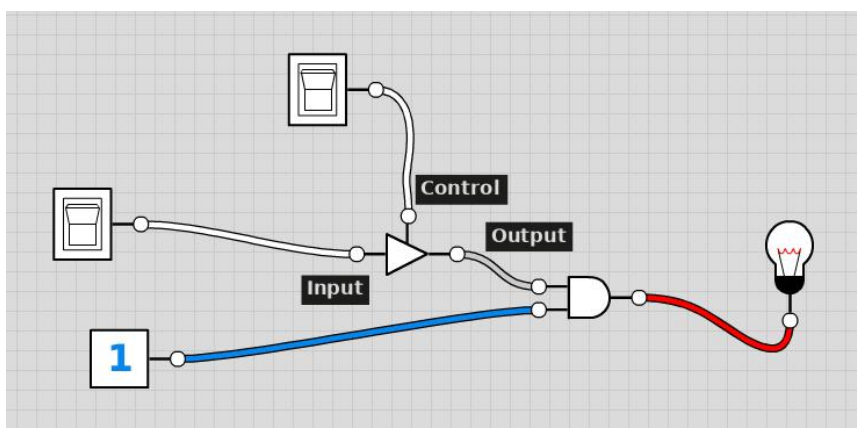
Just by reading the above you should think of a couple of problems. For starters, if the above is true then the bus gets reset everytime it goes through a register, which makes CR A to CR C transfers almost impossible (unless you rely on CR B to temporarily hold the value). The second problem you could think of is how does a register store a value from the bus?



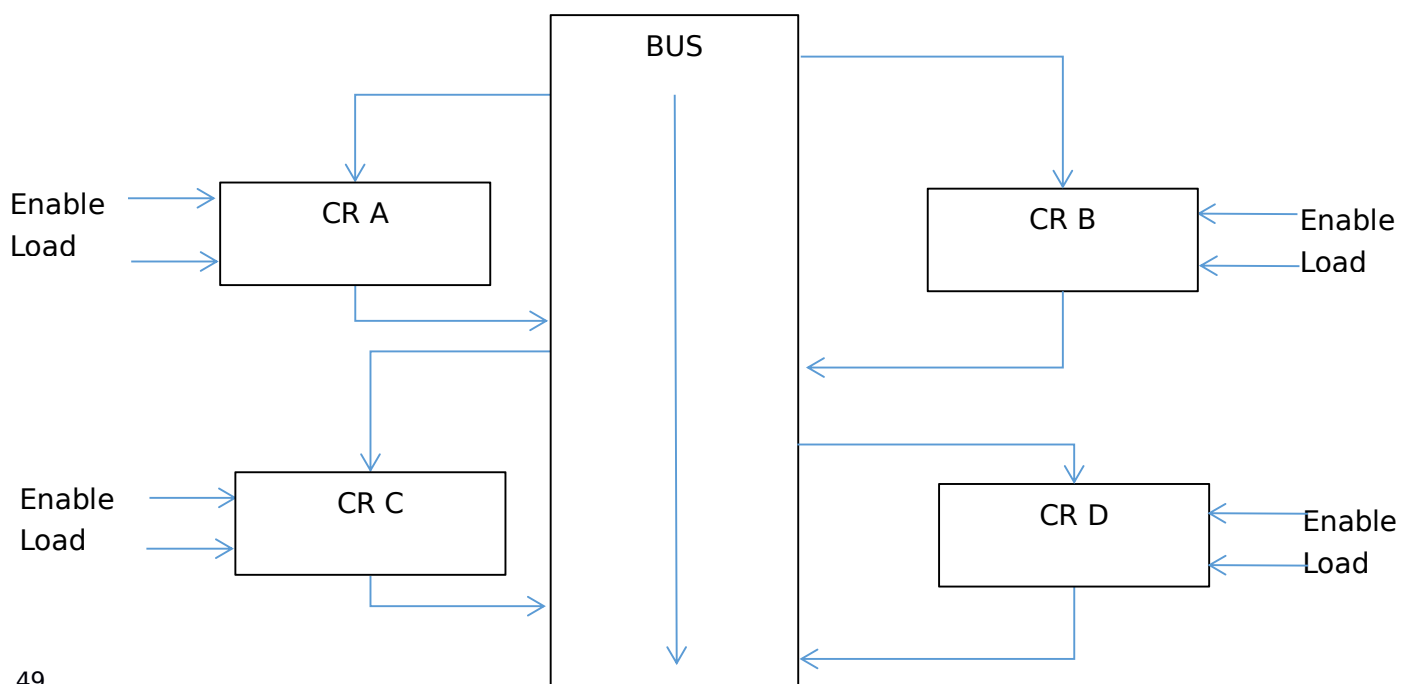
All of this above problems can be solved by adding a tri-state buffer and an Enable and Load setting on our registers. The Enable setting makes whatever is in that register reset the value of the bus to that value. The Load setting loads any value from the bus into the register. This way we can do pretty much anything we want - we can transfer data from CR A to CR C; we can save a value from CR D to CR B etc..etc..

The way we will do this is with the tri-state buffer; the tri-state buffer will allow for the output signal to either make it or not make it to the bus. A wire has the two states 0 or 1, but it can also have a high impedance, Z state. When the Control is 1 in the tri-state buffer the signal output is either 1 or 0; when the Control is 0 the output signal is Z. Below we can see an example and a Truth table for the tri-state buffer.

Input	Control	Output
0	0	Z
0	1	0
1	0	Z
1	1	1



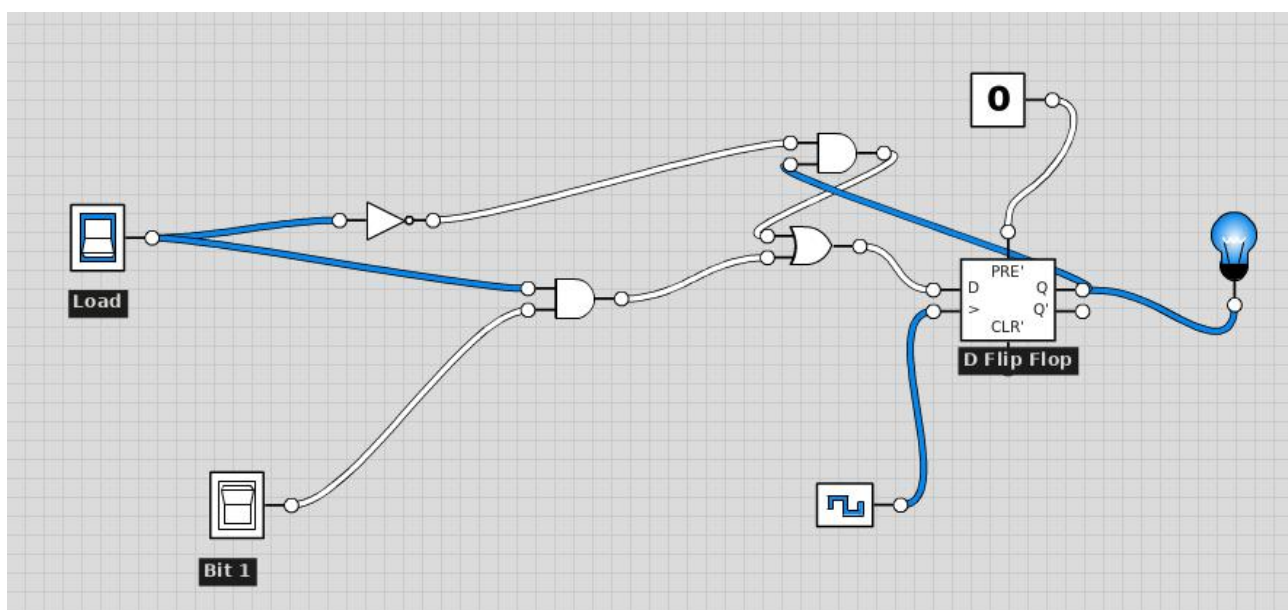
By having a tri-state buffer on both the Enable and the Disable we can make sure that the registers only gather or send data when we, or the logic we will develop later on, desires. An abstract example can be seen here



As we can see now we can have the possibility to make sure something gets loaded or sent through the buses, for example from CR A to CR C the logic behind it would be something along this lines:

<p>Time Passes thanks to our Clock</p>	<p>CR A LOAD = Z ; CR A ENABLE = 1 ; CR DATA = 01011010          CR B LOAD = Z; CR B ENABLE = Z; CR DATA = N/A          CR C LOAD = 1: CR C ENABLE = Z; CR DATA = 01011010</p>
--	--

This only leaves one final question; how can we save the data once its loaded into a register? If we go back to when we were learning about the clock we might remember the D Flip Flop or the SR Latch. By using similar circuits we can store an 8-bit value (until that registers enables the Load again; in which case a new value will be stored)



The above diagram introduces IC to our design process, an IC (Integrated Circuit) is simply an already made circuit condensed into a smaller footprint. The input labeled Bit 1 is the first of 8 bites needed - therefore we will 8 of the above circuits, except for the Clock (the rectangle with the AC current symbol) and the Load input which there should be only one of to insure the whole system follows the same main instruction (Enable ON/OFF) and the same timing (Clock speed).

To build the SAP-1 we will need 2 8-bit common registers and what we will call an instruction register, which greatly resembles the CIR that we discussed previously in the Von Neumann model. Before discussing in more detail the function of the registers please note that in the actual construction of the SAP-1 we will not create 8 of the above circuits. This is because now we know what is inside a single bit register and therefore we know how to create an 8-bit register and therefore we can slightly reduce our workload and need for components by simply using two 4-bit register ICs (sort of what we could do with the clock, instead of building our clock we could just have a clock IC and everything would still work) - they come built and have everything we need for our register-bus communications interface.

Going back to the instruction register; it is simply a copy of the other registers but simply the 4 most significant are not going to go to the bus for data transfer. The 4 most significant values of the instruction register will be linked towards the computer's instruction decoder which will be discussed later on.

0	1	0	1	1	0	1	0
$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$

The diagram above is simply a representation of the 4 most significant bits (shown in a highlighted color)

### SAP-1 CONSTRUCTION: ALU

The ALU as we said before is the part of the processor which is in charge of performing the arithmetic needed for a program to run. When we learned about reading numbers in binary in the 'Electronics' section we said that we would come back to binary for more complicated tasks later on.

The time has arrived to learn about two major things in binary, which our ALU needs to be able to do; adding and subtracting. The way we add numbers has certain logic behind it but it follows our way of doing it. However the way we subtract numbers in binary differs from our typical method. In binary we subtract by converting the second operand of the equation into a negative number and then adding the values together. Lets have an example to understand it better;

Typical method, number A - number B = number C

whereas in binary, number A + negative(number B) = number C

Therefore, and with that in mind, the only thing we need to learn to do is add in binary and convert a positive value to a negative value in binary. To start with we will learn about adding in binary. For all of our examples we will play with 4 bit values that can't carry if the sum overflows (this will be understood later on).

To sum we have to simply follow a set of rules. This set of rules basically tell you whether the result is a 0 or a 1 and whether you have to carry anything into the next bit. Here is the rule table.

Calculation	Result	Carry
0 + 0	0	0
0 + 1	1	0
1 + 0	1	0
1 + 1	0	1
1 + 1 + 1	1	1

You might be wondering why the last entity on this list has three digits in the calculation column and not two, this question will be answered once we do a sum as an example using this table

The sum we will be doing is...

0110  
0111 +

Here we are adding 6 + 7. We start from the right and follow the table and the instructions that it gave us. In the first calculation we result 1 and carry 0

0110  
0111 +  
 1

After writing one as a result and carrying none to the next calculation we move on to the next, now 1 + 1 gives us a result of 0 and a carry of 1

1  
 0110  
0111 +  
 01

So far this has been straight forward, and if we follow the rules table it should stay this way, we know have a carried on 1 which moved on to the next calculation. Doing 1 + 1 + 1 gives us a result of 1 and a carry of 1

11  
 0110  
0111 +  
 101

Now we have the last calculation to do, we treat the carry 1 as if it was another 1 on the next sum and get 1 + 0 which results in a result of 1 and a carry of 0

11  
 0110  
0111 +  
 1101

And that's it, the sum of 0110 + 0111 gives us 1101, which in denary is 6 + 7 giving us 13

Now that we know how to add in binary we need to be able to convert from positive values into denary. To learn this we will look at two models of doing this, the first one, One's complement, will not be used but we need to learn about it to be able to learn about the one which we will use, Two's complement.

One's complement relies on a very simple method. It has the most significant bit used to determine when something is positive and when something is negative, this means that it only leaves 3 bits usable out of a 4 bit value, but it also means that we can represent negative and positive values simply by switching a bit, something incredibly simple to implement in our ALU. Below is an example of a positive number and a negative number under One's complement. Please note, a low most significant bit denotes positive, where as a high most significant bites denotes that the value is negative

One's complement table 4 bit

1000	-7	0001	1
1001	-6	0010	2
1010	-5	0011	3
1011	-4	0100	4
1100	-3	0101	5
1101	-2	0110	6
1110	-1	0111	7
1111	-0		
0000	0		

Straight away we can see that a problem appears, we have a positive zero and a negative zero, this then

when adding the number together (and therefore performing subtractions) creates major problems that make this method completely unreliable. The only one good thing we can take from this method is how we can easily convert numbers from positive to negative by flipping only one bit, the most significant bit.

Therefore and to fix the double zero problem we are going to use another method, this method is called Two's complement and is simply an improvement over One's complement; by subtracting a one to every negative number we eliminate the double zero problem. Two's complement now gives us the appropriate answer when we perform subtraction and additions and is therefore the method we will use in our ALU. To convert a number from Two's complement to One's complement we simply add a one. Below we have a table to show how Two's complement interacts with numbers.

Another good thing about Two's complement is that it allows us to put a header to every column of binary digits, unlike with One's complement - this adds a lot of sense to what we are doing. Below is the table.

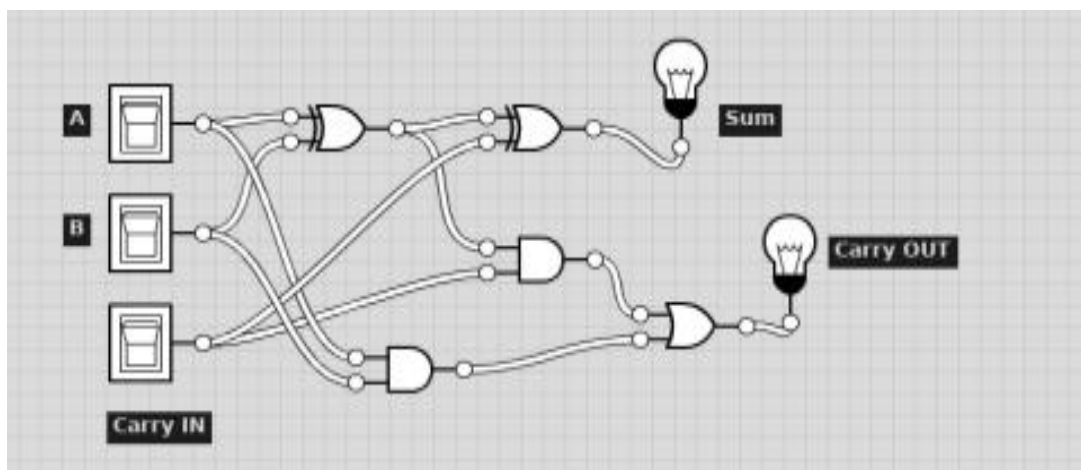
Two's complement 4 bit				
-8	4	2	1	Denary value
1	0	0	0	- 7
1	0	0	1	- 6
1	0	1	0	- 5
1	0	1	1	- 4
1	1	0	0	- 3
1	1	0	1	- 2
1	1	1	0	- 1
1	1	1	1	- 0
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7

Now if we wanted to do the subtraction 3 - 2, we would simply take 3 (0011) and 2 (0010), then convert the 2 into a -2 (1101), after that we would do 3 + -2 and we would get the result. This is how we will perform arithmetic operations in our ALU

Now that we know how to add numbers we need to know how to do a machine to add them. So right now what we need to do is a 4 bit adder. A 4 bit adder is a component which comes in the form of an IC (integrated circuit) but that can also be built from logic gates. We will build a 4 bit adder out of logic gates but on the final SAP-1 build we will use the 2 of the IC adders to have 8 bits. We will also look at how to make the adder interact with the other parts of the computer.

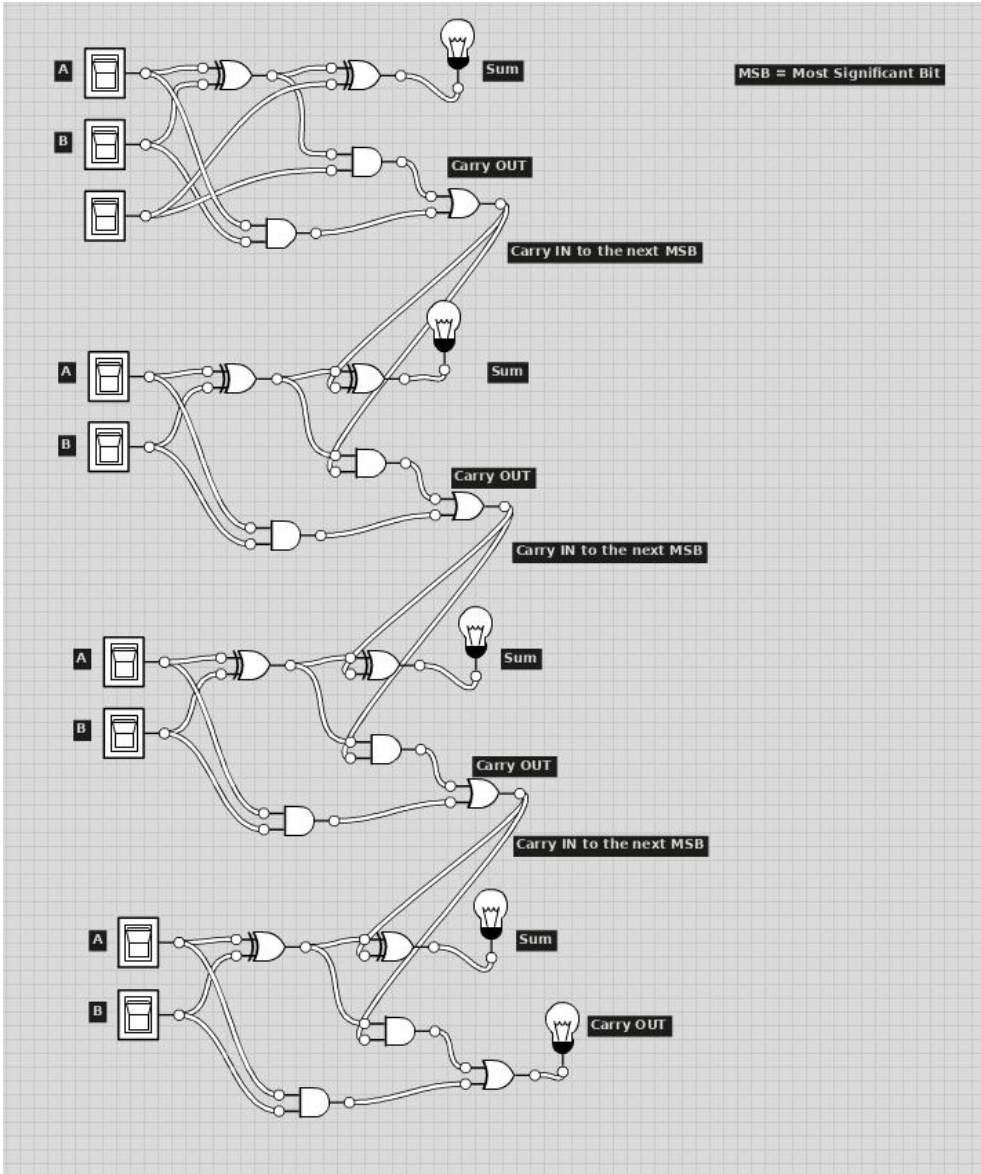
All the components for the adder have already been explained so this should be pretty easy to

understand. First we are going to look at a 1 bit adder.



This right here is a one bit adder. It uses, just like our rules table for addition, a carry value as an output, this carry value jumps to the next calculation thus turning it self from a 0001 to a 0010, which is two in binary, and which is why sometimes we need to have only a carry value and not a result value - because the carry value is 2 and the sum value is 1. And having them together would make a 3.

The adder is fairly simple, having two XOR gates, two AND gates and one OR gate. Each adder that we add handles one bit higher, therefore this adder only handles the 0001s, and if we were to add another adder that adder would handle the 0010s, and the next one would handle the 0100s and so on. The carry OUT output will be linked to the next adder that we add as a way to carry any value. A diagram below, with the help of this 4 bit adder should help you understand this paragraph.



In this example, we have a 4 bit adder, this are 4 one bit adders combined, the way they can 'communicate' or carry each others results is by having the carry OUT connected to the carry IN of the next adder instead of connected to a bulb.

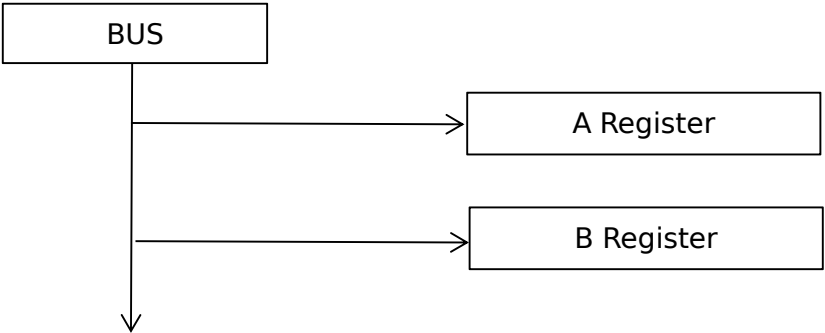
When one adder has a carry as an output we say it is overflown, this means that it has more information than it can handle (they are called one bit adders because they can handle one bit, and if you remember, a carry is always a 2 and not a 1) .

A way of preventing overflows to our adder system is to connect a new adder to the carry OUT of the previous adder, then that second adder will handle the overflown value.

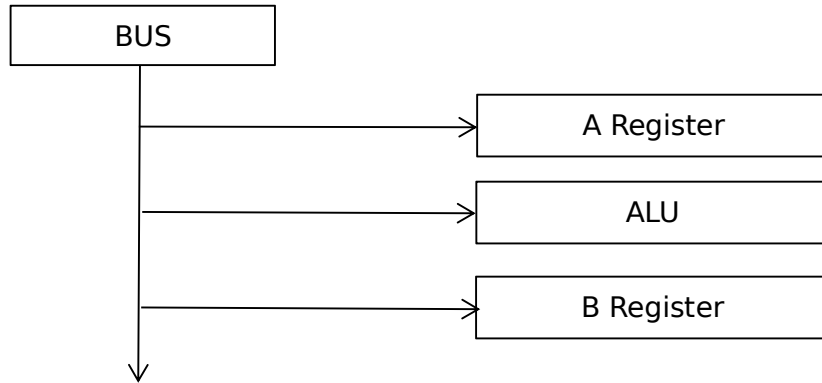
Of course we will still get overflows with a 4bit adder, but at least we'll be able to do more than 1+1 without it breaking

Now that we know how to add and subtract and how to build an adder we need to be able to put it on the computer it self and make sure it interacts with our other components. The components the ALU needs to interact with are mostly the two registers that we have and the bus.

So far we know that we have two Common Registers, we will call them the A register and the B register. They are connected by a bus and that's how they communicate. Here is a visual representation of that.



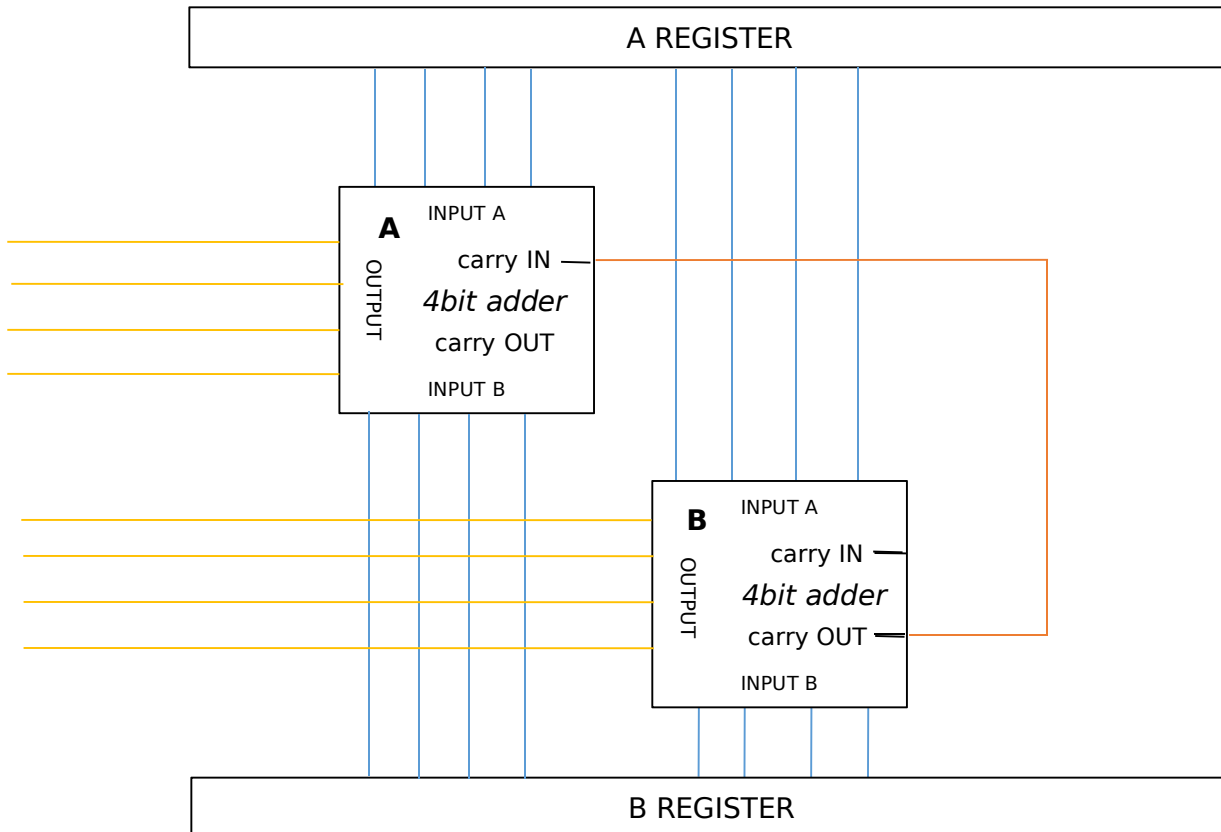
Now we simply add the ALU to this design.



Now that we have the way to communicate we need what inputs and outputs the ALU will have. As an output the ALU will have EO (the E stands for Epsilon, which means sum), and as an input we will have a Subtract signal, or SU, which when high will tell the ALU to subtract and when high will tell the ALU to add.

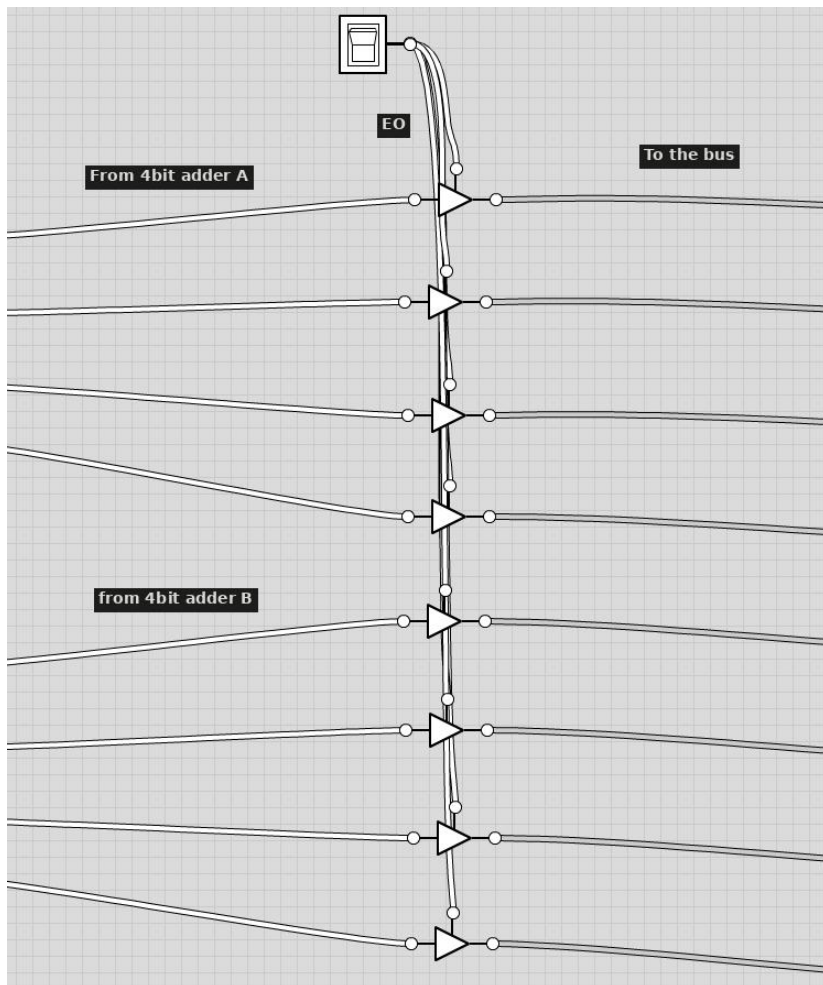


The only thing we need to do know is understand how to connect the ALU to the registers on a less abstract level, the following diagram shows this, each 4bit adder is presented as an IC in the form of a box.





This is how the adder and registers would communicate, if we wanted to then send the result and load them into the bus then we would have to use the same component we used with the A and B registers, a tri-state buffer.



Here we can see the tri-state buffer as the last barrier between having the bus loaded with the output from the ALU.

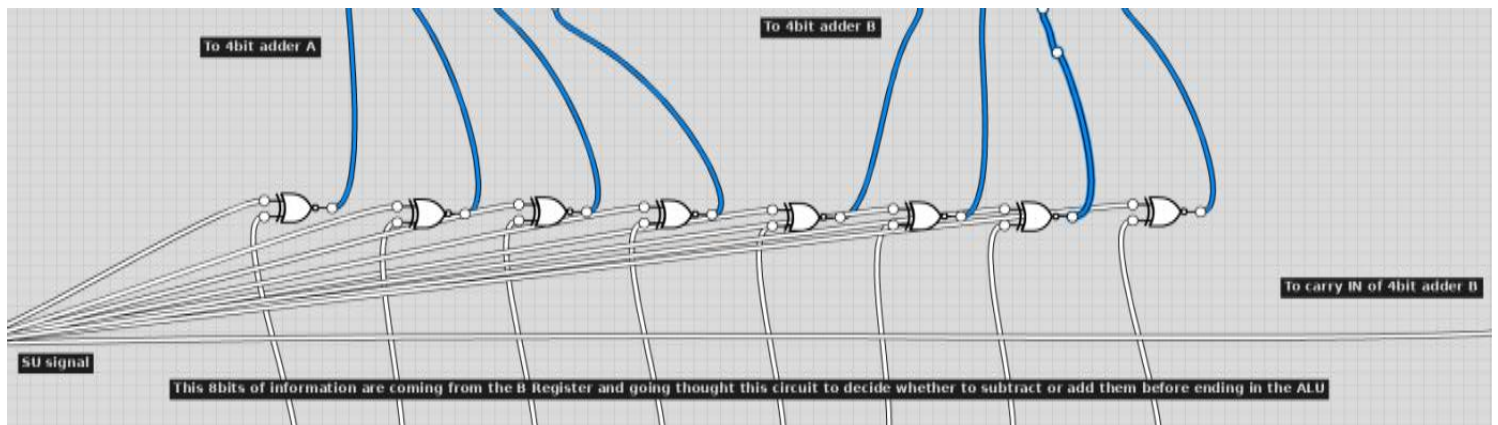
Now the only thing we need to learn is how to subtract and this is surprisingly simple. To subtract we are going to use a logic gate which acts as an inverter when under certain conditions, we can call it a conditional inverter. The XOR is this conditional converter, if we look at the truth table for an XOR gate we will see how under the condition of A being high, then the output is the opposite of B. This is very useful because we now that to subtract we only need to convert a number to negative and add 1 to convert it to its One's complement equivalent. Therefore we can link the B input of the XOR to the SUB signal and therefore be able to add or subtract. Then to convert the number from Two's complement to One's complement we add one by adding one to the Carry IN if the SUB signal is one (meaning that we want to subtract)



Inputs		Output
A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

**EXCLUSIVE OR**

The diagram below will show you how to do the subtraction



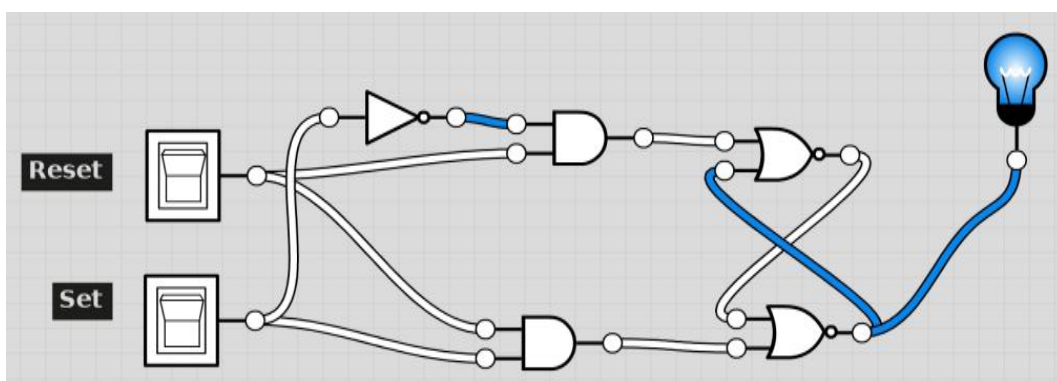
And that is it, that is how we add and subtract numbers in binary, how we lay the ALU within the processor, and how we link it with the bus and registers. This ALU can now perform operations of a value up to 255 before overflowing.

### SAP-1 CONSTRUCTION: RAM

This chapter will not be so much about building but mostly theoretical and abstract content, this is because we have already covered how to build memory. We will look at the two types of RAM and how to build them, then we will look at why one type is better for us than the other one, after that we will look at the method used to address one bit of information without interacting with the other ones and finally we will see how IC technology can be very helpful.

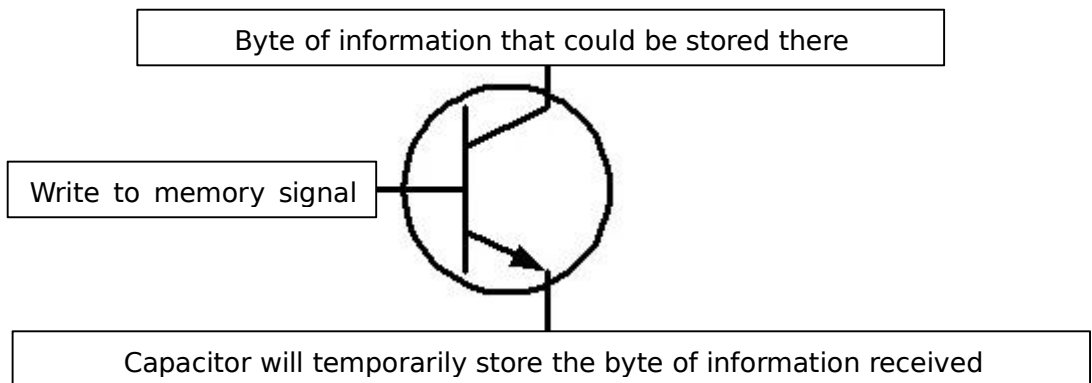
RAM, Random Access Memory, can be simply seen as the memory that we will use in order to store information, like perhaps the result of a calculation. We save results in RAM and not leave them in the bus or registers so that the computer can perform another action. Nowadays, computers come with an average of 6GB of RAM

The first type of RAM we are going to look at is SRAM or Static RAM. This design is based around the use of an SR Latch, just as we had used in the example of the Clock. The circuit will store a value (0 or 1) and then keep it there until the reset signal is sent.

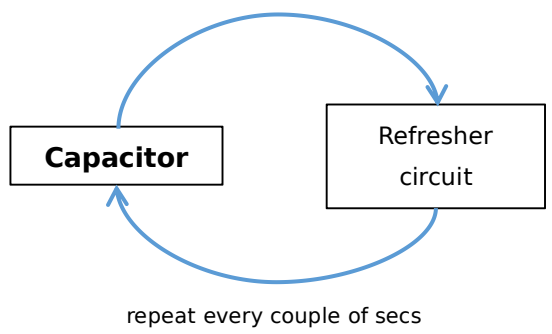


That right there is a way to store one bit of information, but it can be fairly inefficient to make because of all the parts it needs. Counting on the fact that we want our RAM to store 16 bytes of memory, that is 128 of those circuits, which is a total of 512 logic gates. Furthermore, for a modern computer that might have 8GB of RAM, if we were to follow this design then that machine would need about 50 billion logic gates, that is 50,000,000,000!

As we can see that is fairly inefficient, and therefore there was a new design made which was by far more efficient. This new design was called the DRAM or Dynamic RAM. This kind of RAM uses a slightly different approach, but very efficient. The only thing DRAM needs to work is a transistor and a capacitor. The charge on the capacitor stores the byte of information. Just like that we have minimized what we need in terms of components for 16 bytes from 512 logic gates, or about 1000 transistors, to 16 transistors. Here we can see an abstract method of how DRAM works.



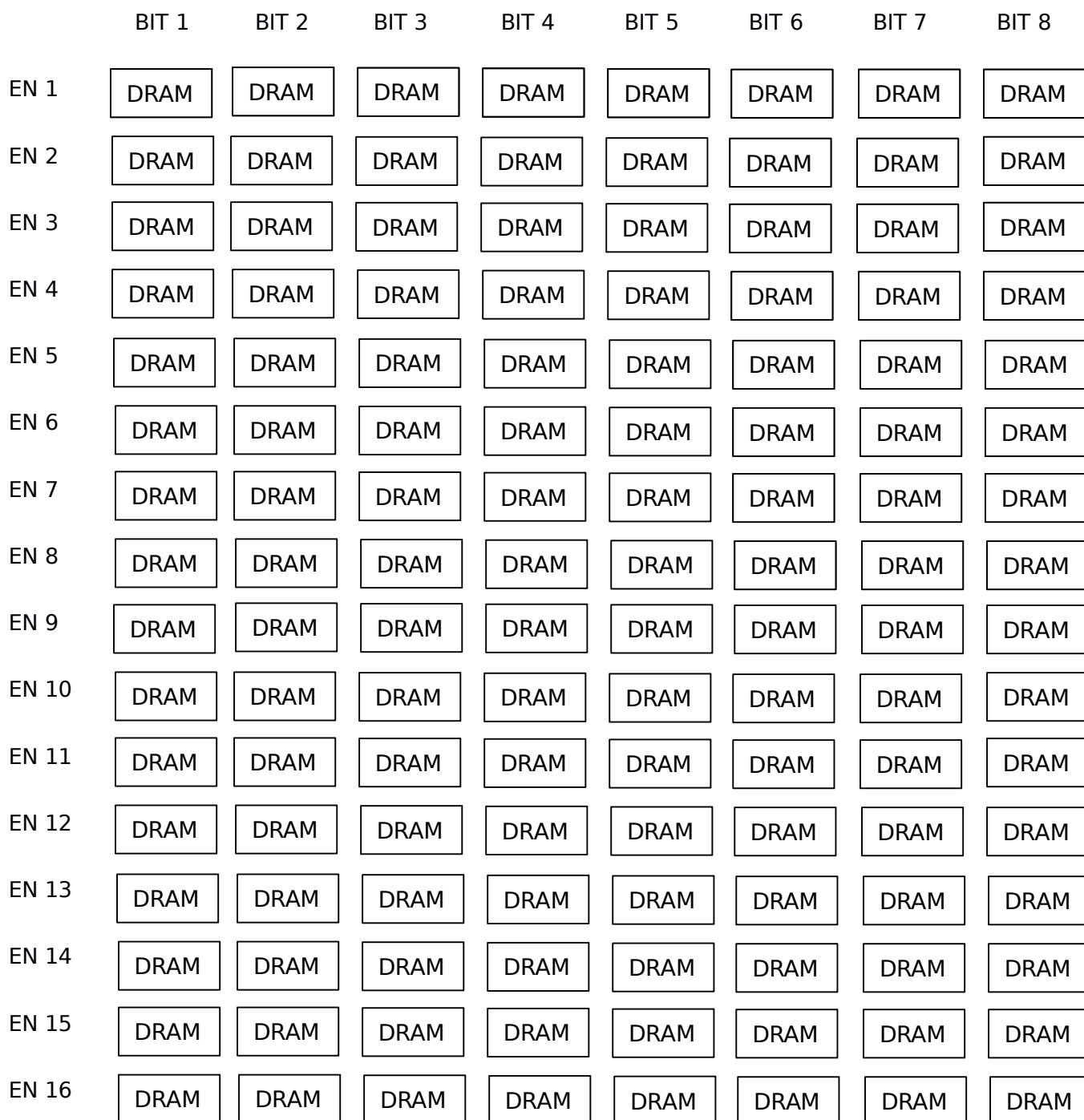
And that is how DRAM works, of course DRAM has a problem; this is because capacitors can only hold a charge for the small time. To fix this the circuit has to be built with another part which acts as an updater. Sort of like this;



Because this is can be more difficult we will use SRAM for the SAP-1 and to explain the next section.

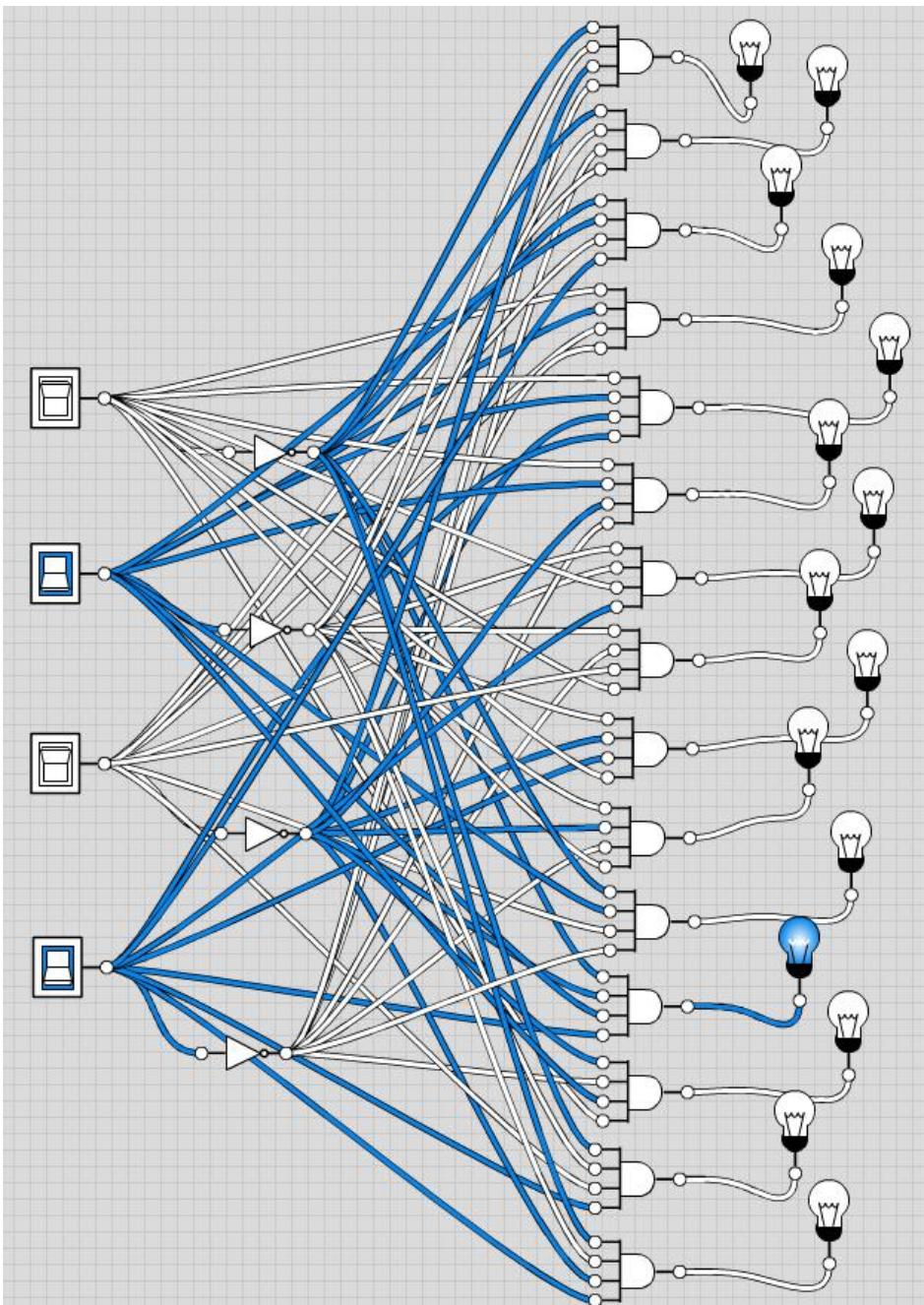
Now we need to look at how to address a certain part of the memory, because we want to load every memory unit of the SRAM with different information. The way we do this is by using 4 inputs, depending on whether they are high or low they will activate one or another AND gate, and depending on which AND gate opens the signal will go to one or another memory circuit.

We can imagine our 128 bit memory to look something like this.



And perhaps sometimes we might want to communicate something to only one EN channel, therefore to fix this we do the AND gate circuit that we discussed previously.





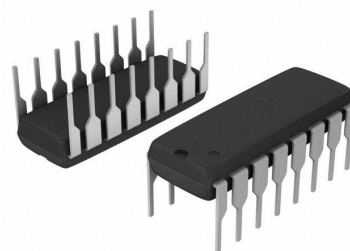
The following is an example on how to address the 16 bytes of SRAM memory, each bulb represents an EN input signal, and therefore depending on which 4bit binary signal is sent down (in the case of the picture its 0101 because of the way the switches are positioned) you can enable one or another memory device.

Overall this circuit is basic but crowded, we vary the input given to the 4-input AND gates and depending on how we change the input we can access any of the 16 registers, presented as light bulbs in this example.

If we can do this for the enable signal (EN), we can also do it for the read signal using a tri-state buffer, with the 3 states being "Read", "Write" or "Do nothing" and just like that we have the ability to have memory in the SAP-1 that can save and deliver information.

However, this task is incredibly long and repetitive, a solution to using so many wires for this memory addressing circuit and so many gates to build the SRAM, can be to simply use a 74LS189 IC. This integrated circuits packs, within its 4 cm of length and 1 cm of height, the whole Read, Write, do Nothing circuit for addressing 64bits of SRAM, so we would only need two of this ICs in order to have the same functionality as we would have with a physical space of about 60m by 60cm. This is one example of how efficient we have become by using microchips.

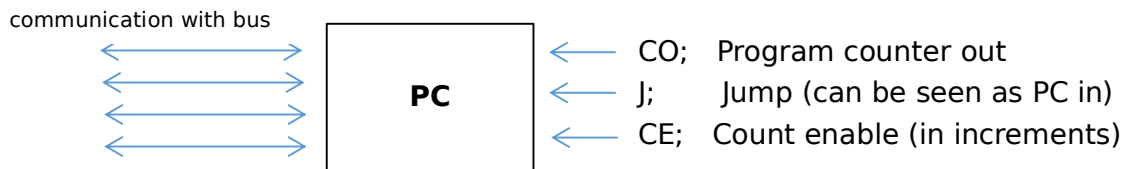
Here we can see the two 74LS189 chips we would need.



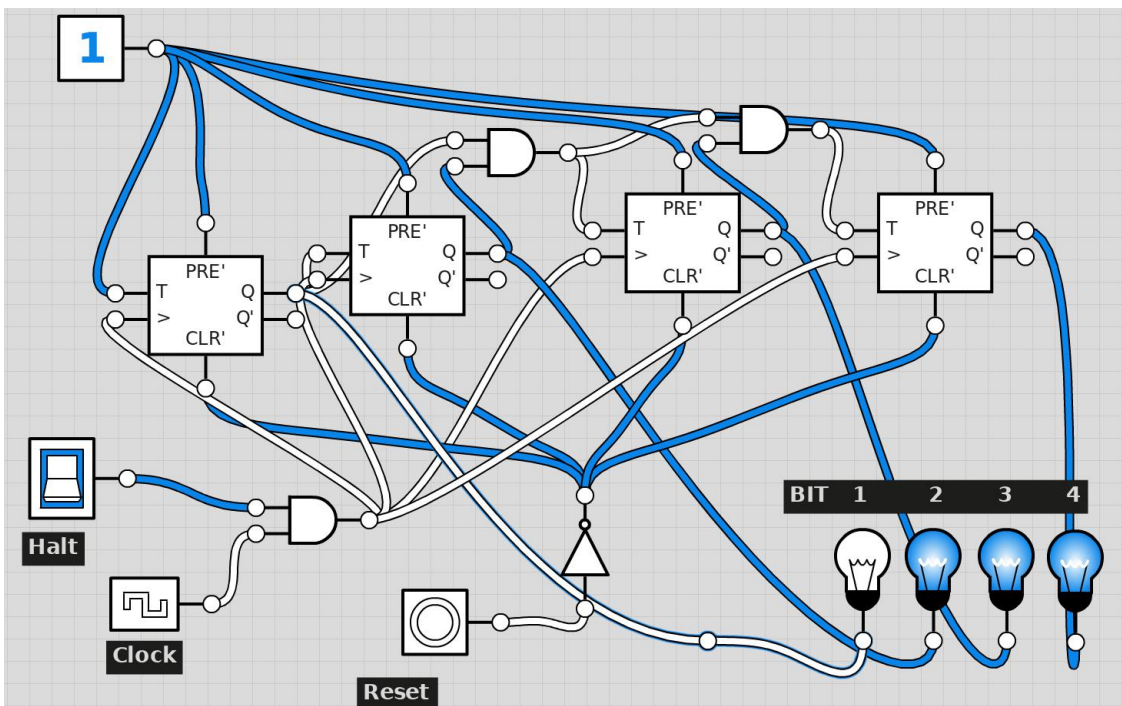
## SAP-1 CONSTRUCTION: PROGRAM COUNTER

The PC is another chapter that has no new electronic fundamentals, but rather one which takes from a variety of the above chapters we've done. The PC must be able to step from instruction to the other one in synchronous, that means in time with the SAP clock. It must also be able to store the next instruction so that that instruction can be sent down the bus to the RAM and the CIR. As one last thing the PC has to be able to step or jump to an instruction, as opposed to always adding one to the current instruction. The jump instruction becomes of use when we want for example to loop through the same program, that is; 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4.... It is then useful to tell the computer once in instruction 4, to jump to instruction 1.

Here we can see an abstract model of what the program counter should be able to do.



As said before, we already know how to store a value and send it somewhere in the bus, we also know how to take input under certain conditions (either tri-state buffer or XOR conditional) and we also know how to count - even though it may seem like we have never learned how to do it, we know the components. By taking an SR Latch, and making it into a single input clock driven device (just like we did with the clock) - this is a simple version of what is known as a JK Flip Flop, and if we combine various of this flip flops by using AND gates as we did in the SRAM addressing circuit, we can come up with a circuit that can count numbers up independently.



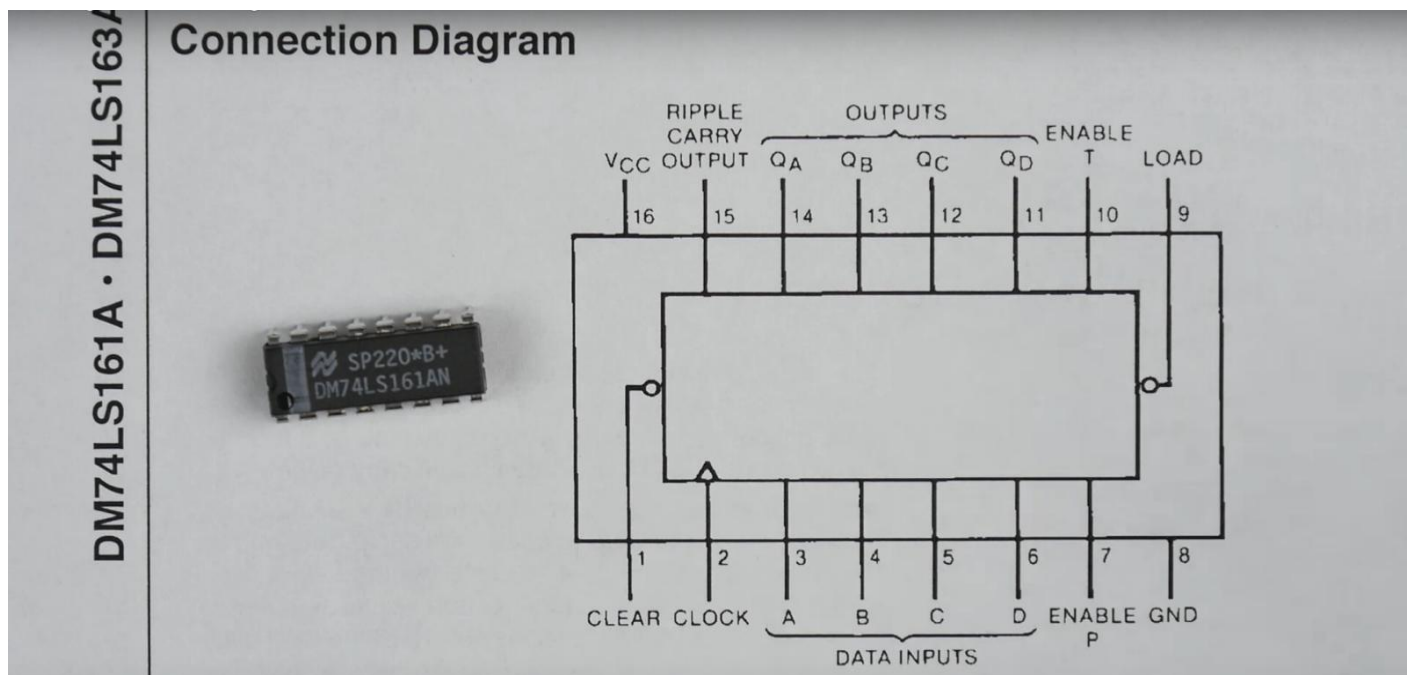
Here we can see how we use 2 dual input AND gates as we are combining 4 circuits.

That is the simplest way to count in binary, and the one we will use for counting in the PC.

Please note, every box represents a single JK Flip-Flop.

Just as it happened with the RAM, we can save the pain from spending an hour worth of building and troubleshooting and just replace the our model with another IC, in this case the DM-4LS161A Synchronous Binary Ripple Counter.

In the section before we just looked at how they look, but in this section and with the aim to learn more we will see how to use them. We are going to look at their schematic (which is the equivalent of looking at a human skeleton). The schematic resides in the component's data sheet, a data sheet, lasting normally from 3 to 10 pages, informs the user of all the information he might want to know about the component.



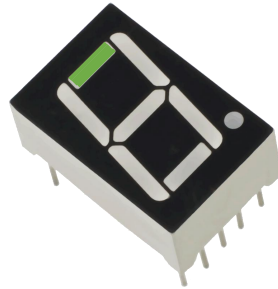
Here we can see the Connection Diagram for our 4bit counter. Now simply we need to know which each pin does to know how to use it instead of building the circuits ourselves like in the page before.

Pin	Title	Description
1	Clear	When high the whole IC will reset
2	Clock	This needs our clock's output as input in order to be synchronous
3	D.I. A	The input needed if we want to jump to a new instruction (for example to loop program)
4	D.I. B	The input needed if we want to jump to a new instruction (for example to loop program)
5	D.I. C	The input needed if we want to jump to a new instruction (for example to loop program)
6	D.I. D	The input needed if we want to jump to a new instruction (for example to loop program)
7	Enable P	This input along with PIN 10 is needed in order for the IC to enable the jump instruction
8	GND	Every IC needs this, by grounding there is a flow of electrons and therefore current.
9	LOAD	This is the output signal when the IC is going to load something to the bus
10	Enable T	This input along with PIN 7 is needed in order for the IC to enable the jump instruction
11	O. QA	Output given to the bus for the CIR, it contains the address of the next instruction
12	O. QB	Output given to the bus for the CIR, it contains the address of the next instruction
13	O. QC	Output given to the bus for the CIR, it contains the address of the next instruction
14	O. QD	Output given to the bus for the CIR, it contains the address of the next instruction
15	R. Carry <sub>out</sub>	This output is connected in to another 74LS161 in case we want to count higher
16	V <sub>cc</sub>	This is simply the 5V in, the pin that powers the IC, flow starts here and ends at GND



## SAP-1 CONSTRUCTION: HEX DECODER

We could have the output of our computer in the form of LEDs, and this would be fine as we know how to read binary. However, and now only thinking about commodity and efficiency, we could have an output display. The component used for this is a 7 segment display, although the name might sound strange they are extremely famous, here is a picture of them



They connect to our output the same way the LED do, but instead of one input they have 7 inputs and we have to use logic to decide when they will go on or off. We are going to do a circuit that will decide when the top section, marked with a small green arrow on the diagram on top, should go on. Before that we need to look at all the possible outputs that we might want our 7-segment display to do.



A 7 segment display can represent 16 numbers, and the "A" and "b" represent 11 and 12, if we were to carry on we would also reach "C", "d", "E" and "F". Now that we know all the possible combinations that the 7 segment display has to offer we are going to take all of the possible numbers and turn them into their binary equivalent, after that we are going to do a truth table based on whether said binary numbers have the top part (shaded in green) on or off. Once that is done we will look work to make a circuit that can manage having the top section on or off depending on the number given (which represents one of the numbers or letters the 7 segment display can do)

First off, lets convert all the numbers and letters into binary

Number	0	1	2	3	4	5	6	7	8	9
Binary	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Number	A	b	C	d	E	F
Binary	1010	1011	1100	1101	1110	1111

Now we can see how each number looks like in binary and we can do the truth table to see if the denary representation of that number has the top part on or off in the 7 segment display, we shall mark the fact that the top part is on as a 1 and as 0 when its off. And we should label the top part as 'a', which is how that segment is called in the component's data sheet.

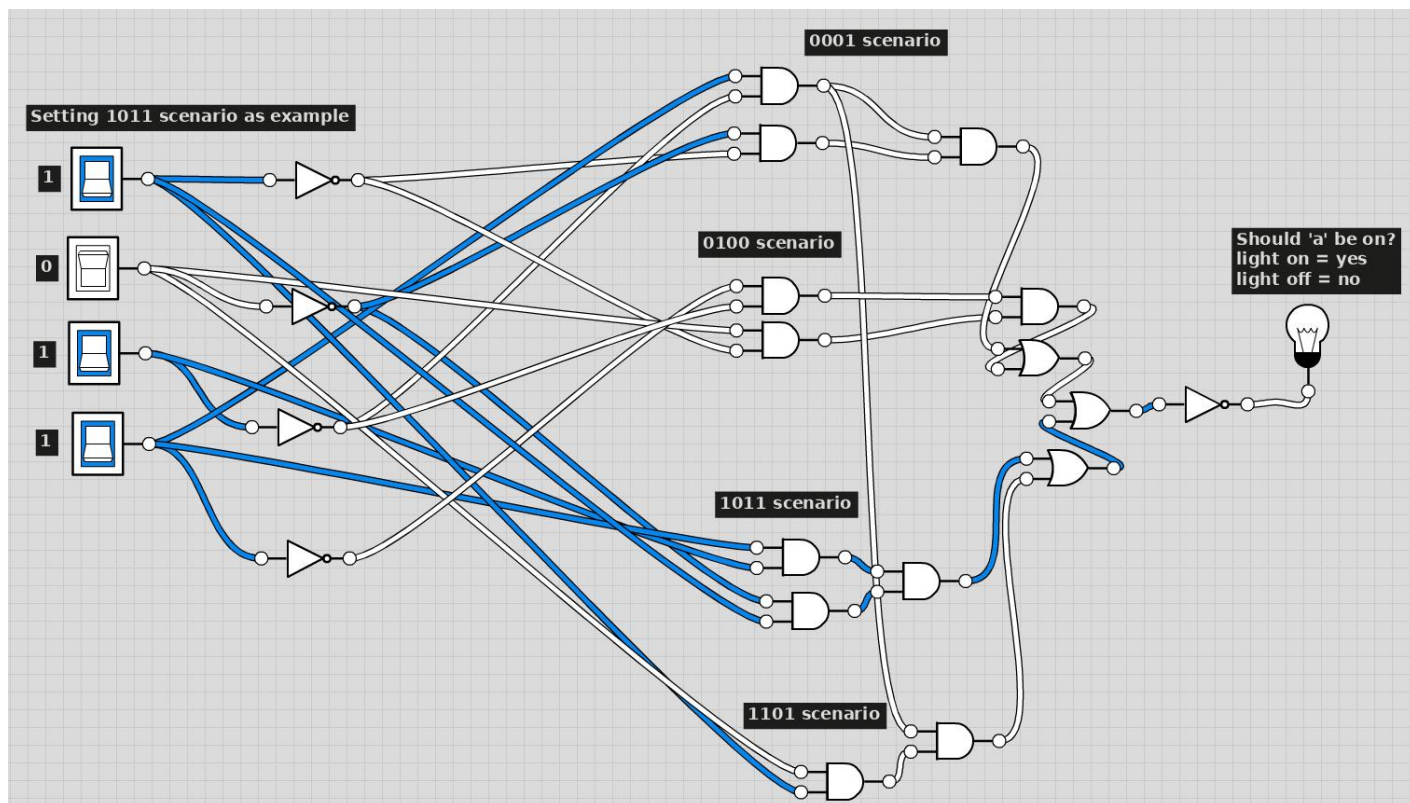


Bit 1	Bit 2	Bit 3	Bit 4	a
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

As we can see here, there are only 4 cases where the top should be off, all the other ones it should be on.

By doing a similar trick to what we did with the RAM addressing we can only 'contact' those 4 cases where the light should be off and tell it to be off, whilst anything else which are not those 4 cases do have the lights on.

This can be done with logic gates. Below we can see one solution. This uses a combination of gates in order to pick out those 4 special cases where the light shouldn't be on, and allows any other input to be on.



Here we see the 1011 as an input, this is one of the inputs that should trigger 'a' to be off, and it indeed does. Now we only need 7 of these conditional logic circuits and we will be able to represent any of the 16 numbers that we have. Of course there is a simpler way of doing this and that is with an EEPROM.

We have seen RAM previously, however there is another type of memory called ROM. ROM stands for Read Only Memory, that means that we can't store things like data or instructions in this memory. We can think of it as a type of memory that holds valuable information for the computer to run. And just as it holds this information it makes it the perfect place to program our functioning of the 7 segment display without the need for the 7 cables and circa 120 logic gates.

Before we get into any more detail, you might be wondering what "EEP" means in "EEPROM". As time went by there have been various revolutions to the ROM and its 'read-only' capabilities. The first one being making the ROM programmable and thus turning it into PROM, this made ROM user programmable instead of coming pre-loaded from the factory, however the user could only program it once. This brought around the EPROM, or the Erasable Programmable Read Only Memory, this allowed the user to, through a special hole in the body of the IC, shine UV rays and erase its memory. However, this method could be very uncomfortable to carry on, as a whole production process would have to be put at halt in order to fix one EPROM or, in the case of hobbyist, he would have to spend hours outside waiting for the sun to deliver enough UV rays as he didn't have the required material to shine artificial UV rays. This brought the EEPROM, which stands for Electronically EPROM. Although the EEPROM is very useful it can be hard to program as it needs special software in order to program it, and therefore we will simply use the longer yet simpler circuit for our output.

### **SAP-1 CONSTRUCTION: FUTURE PROJECTS**

And that is it! A very simple 8bit computer based on the SAP-1 has just been designed, this section will talk about the possible future things that could be done to this computer in order to enhance its functionality, and that I would like to do as I enjoy electronics and this type of projects. The following table could be an example of some nice future additions.

<u>Future addition</u>	<u>Summary</u>
Add an EEPROM for display	By having an EEPROM we would be able to configure many more output displays, which could make the computer more friendly
Program computer on the go	A nice addition could be to extend the use of the instruction register (CIR) which we built simply so that we could talk about its future. By, once again, using an EEPROM, we can program each instruction to do something. For example a 0010 could mean LOAD to CR A from bus.
Print circuit	So far if we were to physically build this circuit it would be done on something called breadboard, or prototyping board. This kind of board, although widely used, is only temporary. One could order this circuit to be printed in a PCB format, which has the cables wiring built into the chassis of the board. This makes it more portable and safer.

Those are just some of the extensions that could be done to this project. But by itself, this is a very capable computer, being able to calculate and save the arithmetical demands equivalent to a young student. Thank you for reading.