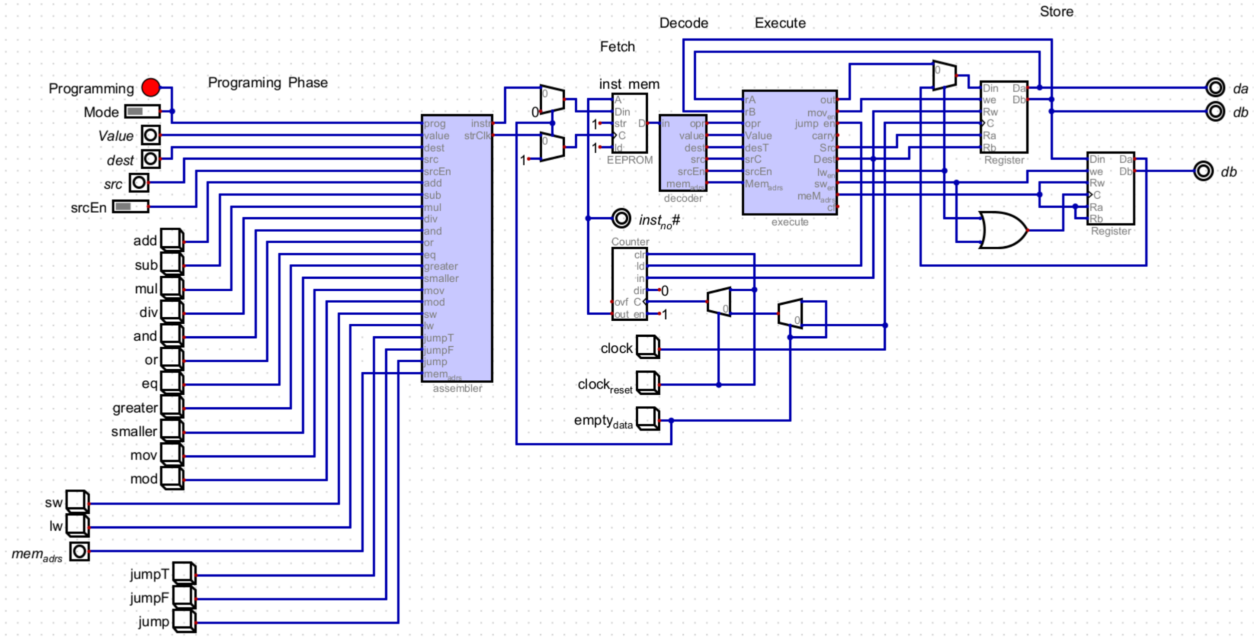
COMP 370

CPU Documentation

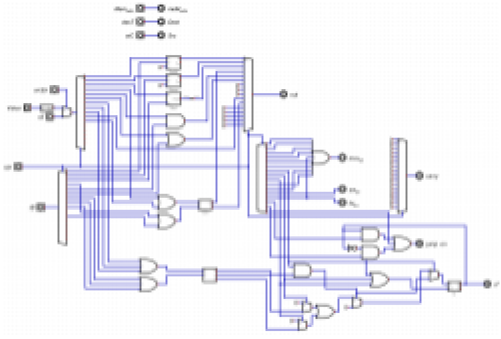
Single Cycle CPU



This CPU is a 32 bit single CPU with two sets of memory, instruction and data memory. The CPU functions in 4 steps: Fetch, decode, execute, and store. There is a programming module in front that allows the instruction to be coded into the instruction memory.

Following the programming section there is a clock system to jump through the memory. This module can function as a clock or memory viewer. If the programming mode is on the instructions are not executed so, you can look at the instructions that have been code and edit them.

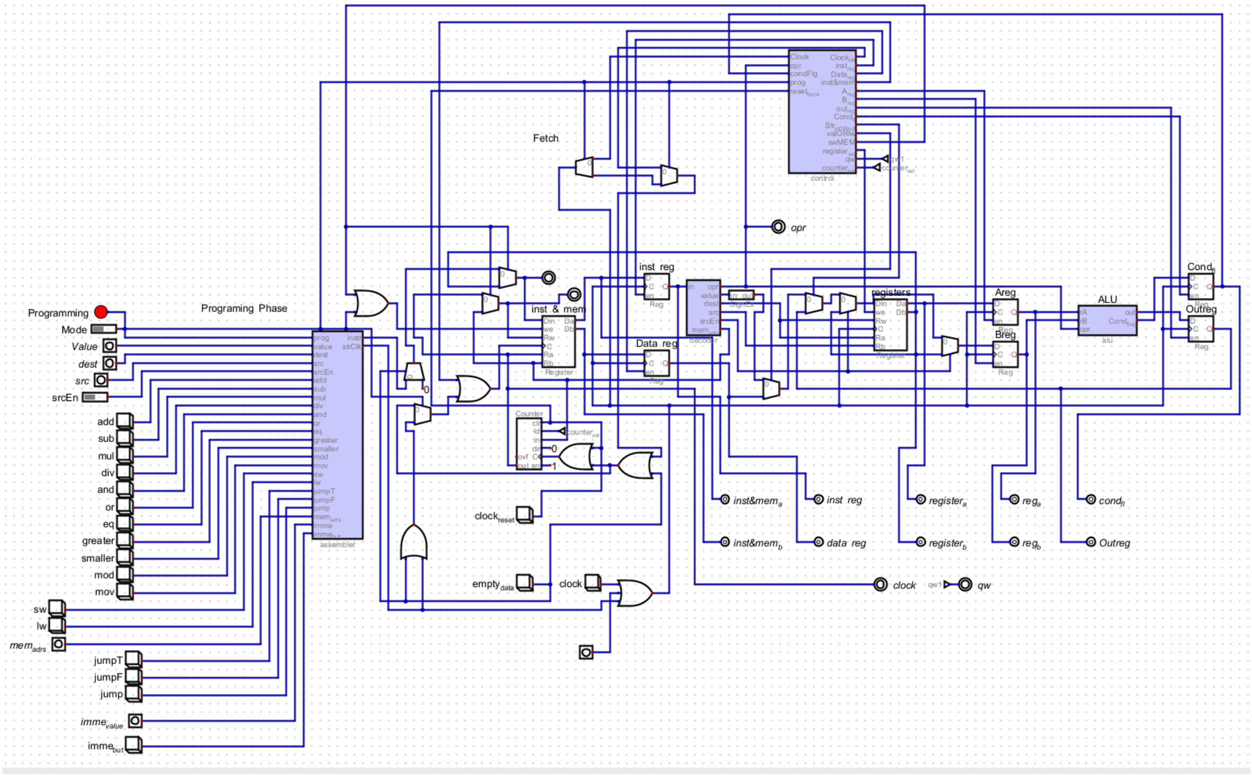
The next main module is the decoder. This is where the instructions are decoded and respective signals are sent to their buses in the execute module. The execute module is composed of the Control and the ALU. Due to this being a single cycle process all the instructions follow the same path so it made more sense to combine the ALU and the Control. The decoder could also have been part of the execute module but the modules were separated according to the main steps of the CPU which are fetch, decode, execute, and store.



Execute Unit

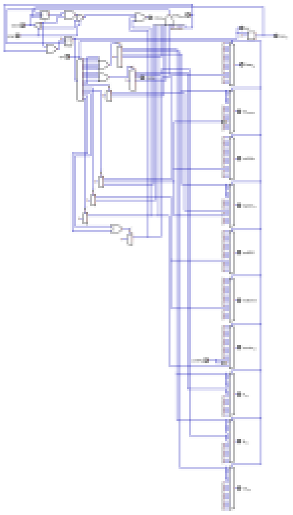
Lastly, the memory modules are part of the last step which is store. Here the execute module either retivies the value from the registers or the data memory. The main differences between the registers and the data memory are that one is permanent and the other one is not, and though here it seems like that both use a similar module but they are representation of two different types of memory. For the temporary use they both use the same simulation component.

Multi Cycle CPU



This is a multi-cycle 32bit cpu. It is capable of all the instructions done by the single cycle CPU. The only major difference between single cycle CPU and the multi-cycle CPU is that the multi-cycle CPU breaks each instruction into different number steps. Given this is a simulation of the CPU the time difference between the calculations is negligible, and on the other hand if they used the same clock to perform the same task, the multi-cycle CPU would perform poorly. The advantages of a multi-cycle CPU are much better shown in the real world where the logic gates have propagation delay. Which makes all the difference between the two types of CPU.

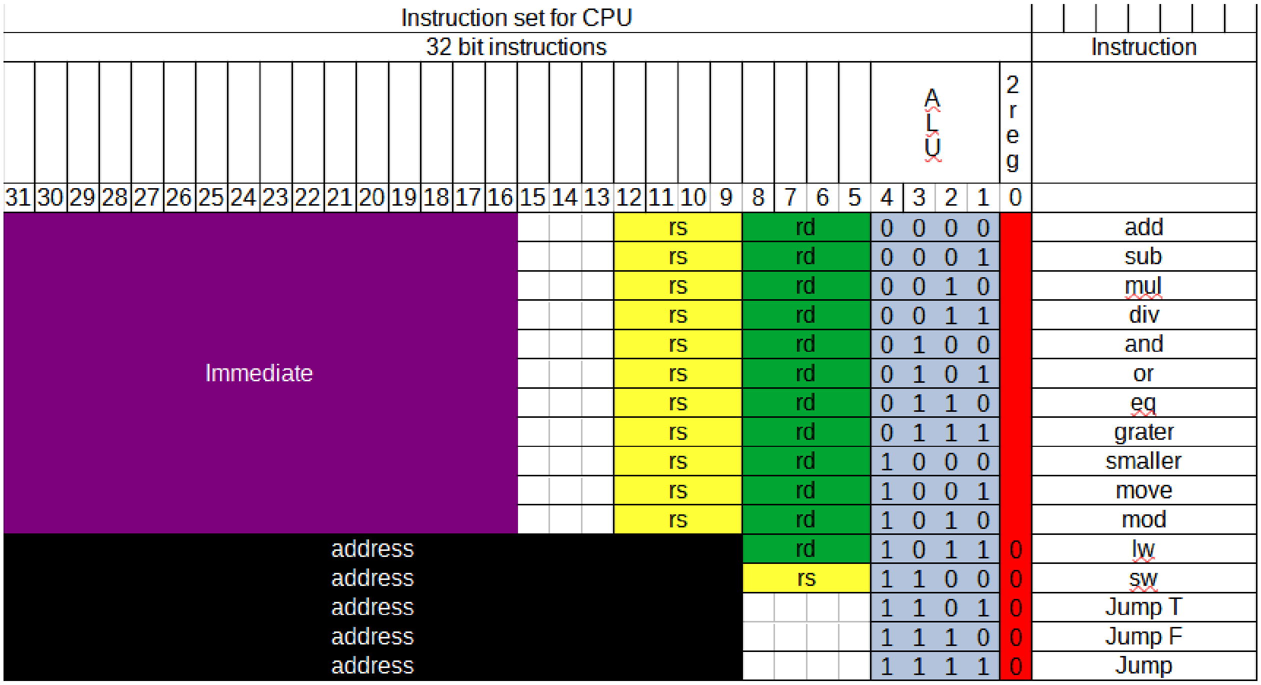
In Single cycle CPU all the steps are identical thus take same time to be executed. Therefore, even a step that can be done in a fraction of the time compared to the other has to be elongated to match the time of other steps. Thus logically wasting time. In multi-cycle CPU this can be avoided by introducing individual stages between the instruction cycles. This way each clock cycle can be used to do the individual stage instead of the entire instruction cycle. Doing one stage at a time gives the CPU the ability to not elongate tasks that only require a fraction of time.



Control Unit

Though it seems easy to explain, introducing these intermediate stages requires a separate and complex Control unit compared to the single cycle CPU. Each and every step is carefully observed and controlled by the control unit. The primary step of all instructions is the same, which is fetch the instructions. So this first clock pulse loads the instruction into the instruction register. Then the second pulse enables the write line in the instruction register. This way we have our instruction to work with. The steps from here on depends on which type of instruction it is. There are 4 types of instruction: Arithmetic, logical, conditional, and data transfer. Arithmetic and logical instruction take the same number of steps but conditional and data transfer instructions depend on the individual instructions.

Instruction set



Instruction set for CPU																																
32 bit instructions																																Instruction
																											ALU				2reg	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Immediate																			rs				rd				0	0	0	0		add
																			rs				rd				0	0	0	1		sub
																			rs				rd				0	0	1	0		mul
																			rs				rd				0	0	1	1		div
																			rs				rd				0	1	0	0		and
																			rs				rd				0	1	0	1		or
																			rs				rd				0	1	1	0		eq
																			rs				rd				0	1	1	1		grater
																			rs				rd				1	0	0	0		smaller
																			rs				rd				1	0	0	1		move
																			rs				rd				1	0	1	0		mod
address																							rd				1	0	1	1	0	lw
address																							rs				1	1	0	0	0	sw
address																											1	1	0	1	0	Jump T
address																											1	1	1	0	0	Jump F
address																											1	1	1	1	0	Jump

Arithmetic instruction:

- Add
- Substract
- Multiply
- Divide
- Module

Logical Instruction:

- And
- Or

Conditional Instruction:

- Equal to
- Greater than
- Smaller than
- Jump if true
- Jump If false

Data Transfer Instructions:

- Move
- Load word
- Store word
- Jump (not a typical data transfer instruction but helps jump when no conditional logic is needed.)

Conclusion

Both the CPUs follow the same instruction set but the inner implementations are slightly different. The components in the simulation do not have any delay between signals traveling thus the logical arrangement of these components is ideal but not realistic. The realistic implementation of these CPUs would highly depend on the components used and now they interact with each other. Nonetheless starting with these ideal logical arrangements is a good start for someone with little knowledge to begin constructing these CPUs.