Operations					128	64	32	16	8	4	2	1	
Immediate	imm				0	0							
Calculate	calc				0	1							
Copy	mov				1	0							
Condition	cond				1	1							
Register 0 in	r0i						0	0	0				
Register 1 in	r1i						0	0	1				
Register 2 in	r2i						0	1	0				
Register 3 in	r3i						0	1	1				
Register 4 in	r4i						1	0	0				
Register 5 in	r5i						1	0	1				
Input	in						1	1	0				
Unused							1	1	1				
Register 0 out	r0o									0	0	0	
Register 1 out	r1o									0	0	1	
Register 2 out	r2o									0	1	0	
Register 3 out	r3o									0	1	1	
Register 4 out	r4o									1	0	0	
Register 5 out	r5o									1	0	1	
Output	out									1	1	0	
Unused										1	1	1	
Or	or									0	0	0	
Nand	nand									0	0	1	
Nor	nor									0	1	0	
And	and									0	1	1	
Add	add									1	0	0	
Sub	sub									1	0	1	
Unused										1	1	0	
Unused										1	1	1	
Never	never									0	0	0	
Equal to 0	jeq									0	0	1	
Less than 0	jlt									0	1	0	
Less than or Equal to 0	jle									0	1	1	
Always	jmp									1	0	0	
Not Equal to 0	jne									1	0	1	
Greater than or Equal to 0	jge									1	1	0	
Greater than 0	jgt									1	1	1	

Instruction Layouts

	128	64	32	16	8	4	2	1
Copy	Operation		Copy from			Copy to		
Calculation	128	64	32	16	8	4	2	1
	Operation		Unused			Condition		
Condition	128	64	32	16	8	4	2	1
	Operation		Unused			Function		
Immediate	128	64	32	16	8	4	2	1
	Operation		Value					

Info + Setup

"Immediate" is the term for sending a value directly to reg0. The immediate value is marked in the bits 1-6 of the instruction byte.
An immediate value can range from 0 - 63

Calculations always take the values in reg1 and reg2, and do the function on them. Reg1 is on the left of the calculation
Example: Addition = reg1 + reg2, Subtraction = reg1 - reg 2

Conditions always take the value in reg3 and compare it against the given condition.
If the condition is evaluated as true, the program counter will jump to the line of code at the value stored in reg0.

When programming, the furthest left bit is the 128th bit. To program a component, simply replace low constants with high constants and rewire.
When finished, package the whole circuit as one component and copy and paste into the processor file.
When you have added it into the file, rewire it in place of the placeholder program component.

Setup

Toggle the PRE and CLR switches by the program component and replace the program component with your own program,
It is not recommended to use a clock in the processor, but rather it should be hand-pulsed.
Nearby to the program component - behind the counter- will be a switch connected to the clock input of the counter.
When pulsing the clock, always wait for a small amount of time before executing the next instruction, to ensure that there are no accidental errors.
Change the values of the input section if required, nearby to the output.
Enjoy!

Assembly Code Syntax

-- Using the Replit file		mov:	mov (from adr) (to adr)			*more similary to cpy					
		calc / cond:	calc / cond (operation)								
		imm:	imm (value)								
		Conditions are slightly tricky to implement. Because the jump address is always stored in r0, you must send an immediate value to r0 before the jump.									
		The immediate value should be the byte of code that you want to jump to, e.g. byte 3, byte 2, etc.									
		Your lines of code begin at byte 0, and increment by 1 for each instruction.									
		After the immediate instruction, you then add your jump condition, e.g. cond jne.									