Department of Electronic and Telecommunication Engineering
University of Moratuwa

Processor Design

EN3030: Circuits and Systems Design

Team CircodIT

Name	Index No:
AMARATHUNGA H.G.N.P.	190037D
PERERA W.A.L.P.D.	190461V
RAJAPAKSHA R.M.P.A.P.	190484T
RANASINGHE K.K.H.	190494A

This report is submitted in partial fulfillment of the requirements
for the module EN3030 Circuits and Systems Design.

12th of Feb 2023

Contents

1. Introduction

This project aims to build a single-cycle RISC-V 32-bit CPU (Central Processing Unit) with a victim cache, simulate it using the Verilog hardware description language, and eventually implement it in hardware using a programmable logic device such as an FPGA (Field Programmable Gate Array). This report explains the design of the microprocessor and CPU, as well as the test scripts required to validate it and the physical hardware implementation.
A microprocessor is a type of computer processor that combines the functionality of a computer's central processing unit (CPU) on a single integrated circuit (IC) or a few integrated circuits. A microprocessor is a multifunctional, clock-driven, register-based programmable electronic device that accepts digital or binary data as input. It processes using instructions stored in its memory and outputs the results.
Microprocessors contain both combinational logic and sequential digital logic.
Microprocessors operate on numbers and symbols represented in the binary numeral system.
Microprocessors are now found in thousands of objects that were previously unrelated to computers. Large and small household appliances, automobiles (and their additional equipment units), vehicle keys, tools and test instruments, toys, light switches/dimmers and electrical circuit breakers, smoke alarms, battery packs, and hi-fi audio/visual components are examples of these (from DVD players to phonograph turntables). Cell phones, DVD video systems, and HDTV broadcast systems, for example, all necessitate consumer devices with strong, low-cost microprocessors. Increasingly strict pollution control rules effectively oblige automotive manufacturers to utilize microprocessor engine management systems to enable optimal control of emissions over a wide range of operating situations. To obtain the outcomes available with a microprocessor, non-programmable controls would require complex, bulky, or expensive implementation.
As a result, we may argue that microprocessors are critical electronic devices for implementing modern-day applications. Because all modern-day applications rely on microprocessors, many types of microprocessors have been developed by various companies to do certain jobs. As a result, developing and implementing a specific microprocessor to do a certain task is critical. In this project, we want to develop and implement such a microprocessor to do certain tasks as efficiently as possible.

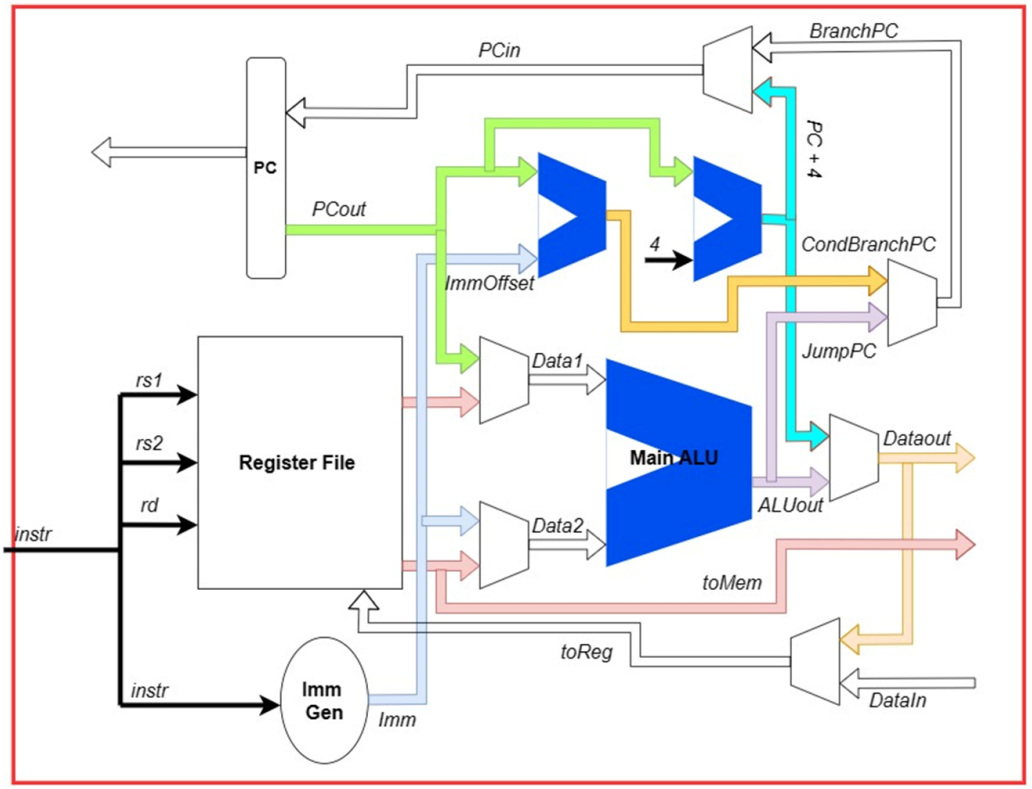
2. Instruction set architecture

a. General architecture

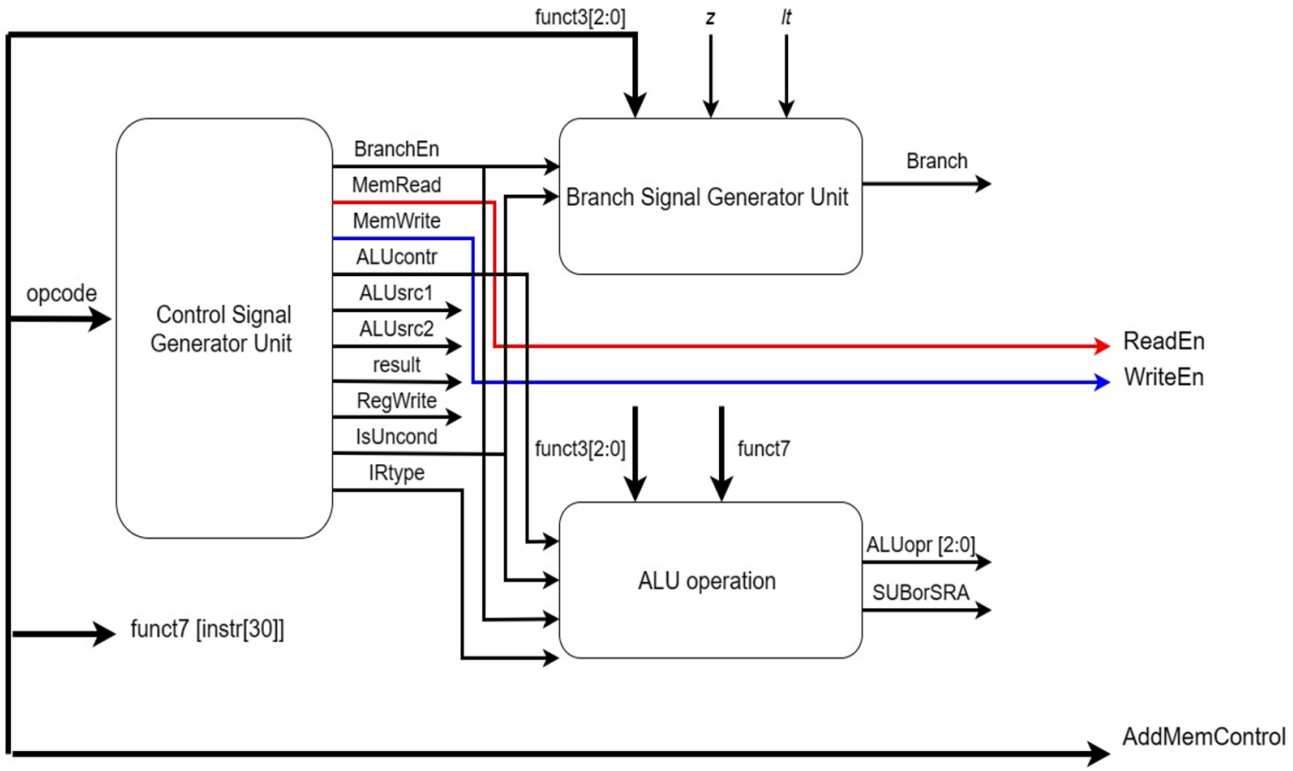
Any implementation of the RISC-V instruction set architecture (ISA) must include a base integer ISA (instruction set architecture), as well as any optional base ISA extensions. The early RISC processors' base integer instruction set architecture is extremely similar to it, with the exception of the absence of branch delay slots and the ability for optional variable-length instruction encodings. The base is rigorously constrained to a limited set of instructions that is adequate to serve as a viable target for operating systems, compilers, assemblers, and linkers. The width of the integer registers and the associated size of the user address space

defines each base integer instruction set. There are two main base integer variants: RV32I and RV64I, which offer user-level address spaces that are 32 bits or 64 bits, respectively. Fixed-length 32-bit instructions in the fundamental RISC-V ISA must be naturally aligned on 32-bit boundaries. Little-endian memory is used in the fundamental RISC-V ISA.

b. Data path



c. Control signals



d. Instruction set

RV32I Base Instruction Set

imm[31:12]				rd	0110111	LUI
imm[31:12]				rd	0010111	AUIPC
imm[20|10:1|11|19:12]				rd	1101111	JAL
imm[11:0]		rs1	000	rd	1100111	JALR
imm[12|10:5]	rs2	rs1	000	imm[4:1|11]	1100011	BEQ
imm[12|10:5]	rs2	rs1	001	imm[4:1|11]	1100011	BNE
imm[12|10:5]	rs2	rs1	100	imm[4:1|11]	1100011	BLT
imm[12|10:5]	rs2	rs1	101	imm[4:1|11]	1100011	BGE
imm[12|10:5]	rs2	rs1	110	imm[4:1|11]	1100011	BLTU
imm[12|10:5]	rs2	rs1	111	imm[4:1|11]	1100011	BGEU
imm[11:0]		rs1	000	rd	0000011	LB
imm[11:0]		rs1	001	rd	0000011	LH
imm[11:0]		rs1	010	rd	0000011	LW
imm[11:0]		rs1	100	rd	0000011	LBU
imm[11:0]		rs1	101	rd	0000011	LHU
imm[11:5]	rs2	rs1	000	imm[4:0]	0100011	SB
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	SH
imm[11:5]	rs2	rs1	010	imm[4:0]	0100011	SW
imm[11:0]		rs1	000	rd	0010011	ADDI
imm[11:0]		rs1	010	rd	0010011	SLTI
imm[11:0]		rs1	011	rd	0010011	SLTIU
imm[11:0]		rs1	100	rd	0010011	XORI
imm[11:0]		rs1	110	rd	0010011	ORI
imm[11:0]		rs1	111	rd	0010011	ANDI
0000000	shamt	rs1	001	rd	0010011	SLLI
0000000	shamt	rs1	101	rd	0010011	SRLI
0100000	shamt	rs1	101	rd	0010011	SRAI
0000000	rs2	rs1	000	rd	0110011	ADD
0100000	rs2	rs1	000	rd	0110011	SUB
0000000	rs2	rs1	001	rd	0110011	SLL
0000000	rs2	rs1	010	rd	0110011	SLT
0000000	rs2	rs1	011	rd	0110011	SLTU
0000000	rs2	rs1	100	rd	0110011	XOR
0000000	rs2	rs1	101	rd	0110011	SRL
0100000	rs2	rs1	101	rd	0110011	SRA
0000000	rs2	rs1	110	rd	0110011	OR
0000000	rs2	rs1	111	rd	0110011	AND
0000 | pred | succ		00000	000	00000	0001111	FENCE
0000 | 0000 | 0000		00000	001	00000	0001111	FENCE.I
000000000000		00000	000	00000	1110011	ECALL
000000000001		00000	000	00000	1110011	EBREAK
csr		rs1	001	rd	1110011	CSRRW
csr		rs1	010	rd	1110011	CSRRS
csr		rs1	011	rd	1110011	CSRRC
csr		zimm	101	rd	1110011	CSRRWI
csr		zimm	110	rd	1110011	CSRRSI
csr		zimm	111	rd	1110011	CSRRCI

3. Register File

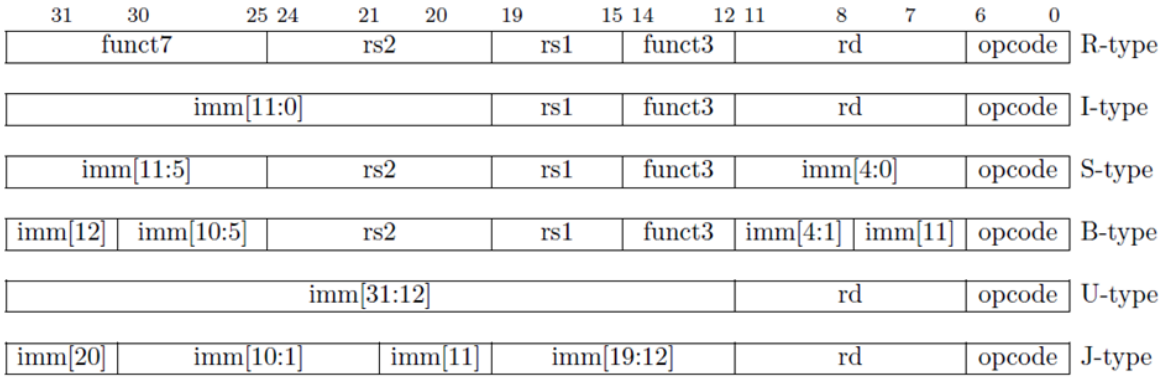
There are 32 registers, x0-x32 in the register file. Integer values are stored in those general-purpose registers x1-x31. The constant 0 is hardwired into register x0. However, the typical software calling convention uses register x1 to hold the return address on a call since there isn't a hardwired subroutine return address link register. The x registers for RV32 are 32 bits wide.

Reg	ABI/Alias	Description	Saved
x0	zero	Hard-wired zero	
x1	ra	Return address	
x2	sp	Stack pointer	yes
x3	gp	Global pointer	
x4	tp	Thread pointer	
x5	t0	Temporary/alternate link register	
x6-7	t1-2	Temporaries	
x8	s0/fp	Saved register/frame pointer	yes
x9	s1	Saved register	yes
x10-11	a0-1	Function arguments/return value	
x12-17	a2-7	Function arguments	
x18-27	s2-11	Saved registers	yes
x28-31	t3-6	Temporaries	

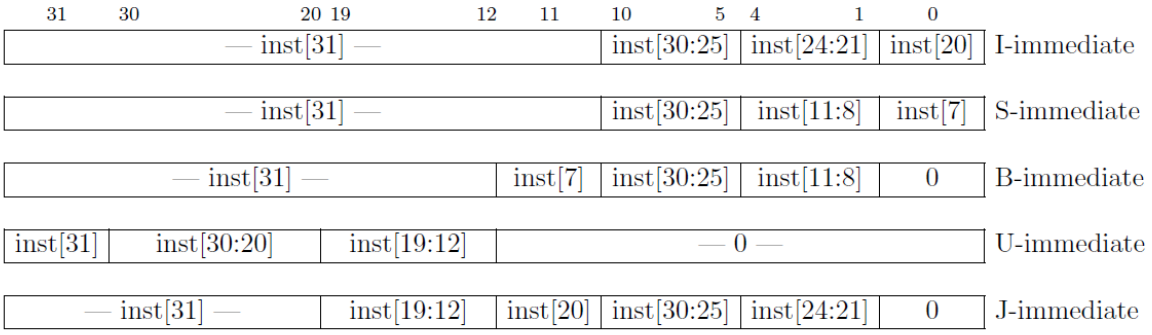
4. Immediate Generator

The immediate generator in RISC-V 32-bit CPUs is a hardware unit that generates immediate values used in arithmetic and logical instructions. Immediate values are small constants embedded in the instruction itself, rather than loaded from memory, which can reduce memory access and instruction fetch overhead. The immediate generator can produce different types of immediate values, including sign-extended and zero-extended values, shifted values, and compressed values, depending on the specific instruction format and encoding. The immediate generator is an essential component of RISC-V CPUs and helps to improve performance and reduce power consumption.

The output of the immediate generator in RISC-V 32-bit CPUs can vary based on the input instruction type. Different types of instructions have different formats and encoding schemes, which affect how the immediate value is generated. So, the RISC-V base instruction formats immediate variants and the output of the immediate generator according to the instruction type is shown below.



RISK-V basic instruction type formats



Types of immediate produced by RISC-V instructions.

5. Instruction memory and program counter

The program counter (PC) and instruction memory are two closely related components in the operation of a CPU. The program counter is a register that holds the memory address of the next instruction to be executed by the CPU. The instruction memory is a type of memory that stores the machine code instructions that the CPU executes. The size of instruction memory of our design is 32x32 bits.

The program counter is used to fetch instructions from the instruction memory in a sequential order. When the CPU is executing a program, the program counter holds the address of the current instruction, and the CPU uses this address to fetch the next instruction from the instruction memory. The instruction is then stored in an instruction register before being decoded and executed by the CPU.

As each instruction is executed, the program counter is updated to point to the next instruction in memory. This process continues until the program is complete or a branch or jump instruction is encountered, which causes the program counter to be updated to a different address.

6. Data Memory

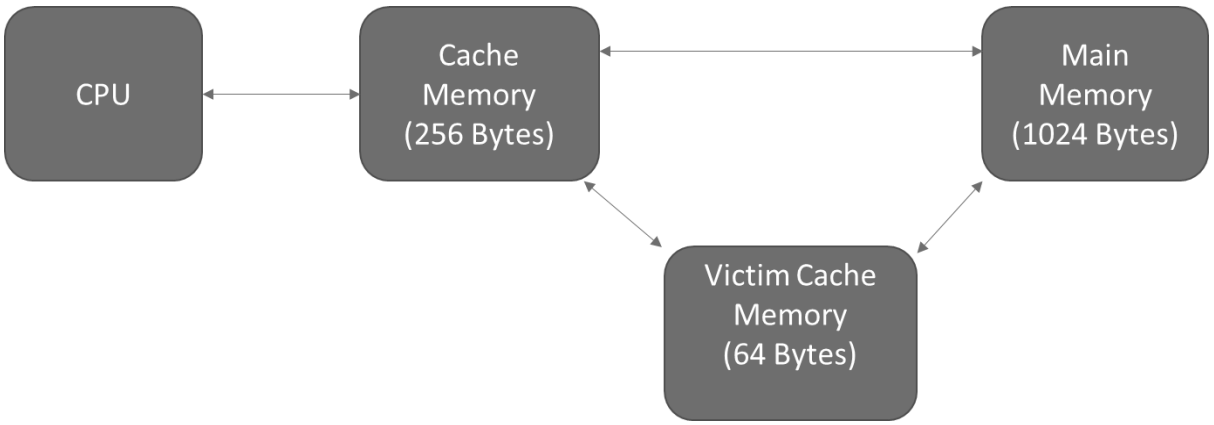
a. Memory architecture

In computer architecture, data memory refers to the physical storage devices used to store data and instructions for a computer system. In order to improve the performance of data access, data memory is usually equipped with a cache and a victim cache.

A cache is a small and fast memory that acts as a buffer between the main memory and the processor. When a processor requests data from main memory, the cache checks if the data is stored in it. If the data is found in the cache, it is returned to the processor, which results in faster data access. This is called a cache hit. If the data is not found in the cache, it is retrieved from main memory and stored in the cache for future use, which is called a cache miss.

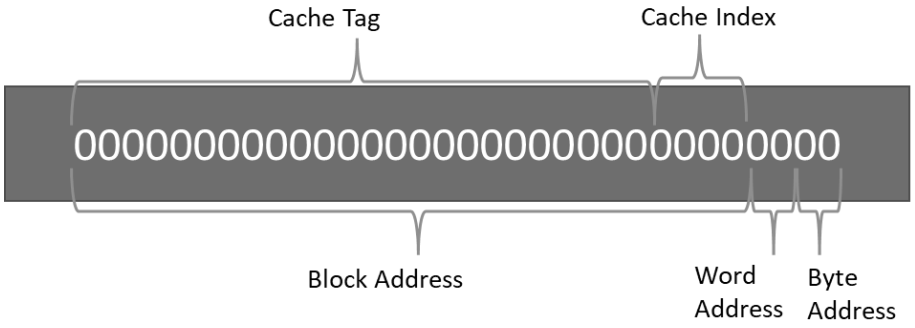
A victim cache is a smaller and slower cache that acts as a backup for the main cache. When the main cache is full and a new piece of data needs to be stored, the cache controller evicts an old piece of data and stores it in the victim cache. The purpose of the victim cache is to provide a temporary holding place for evicted data so that it can be quickly retrieved if the same data is requested again in the near future.

In summary, the cache and the victim cache work together to improve the performance of data memory access by providing faster data retrieval and temporary storage for frequently accessed data.
Both cache and victim cache implemented with direct mapping and write through policy is use as write policy in this cache architecture.

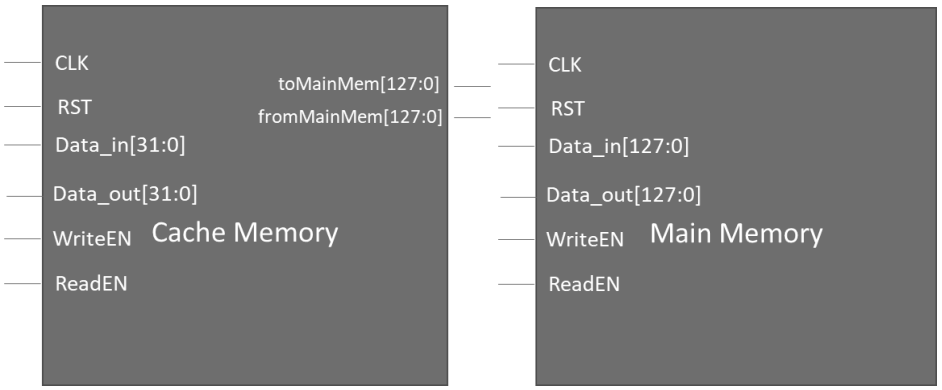


b. Memory Address

Memory address in this architecture is byte addressable and it has 32 bit width. In the main memory, word size is 32bit and 4 words included into one memory block. There are 64 memory blocks in the main memory. So block address should include 6bits and 10 bits required to represent a byte in memory. Theoretically maximum 4GB memory can be used with this architecture. But to make this simplified take it as 1024 bytes.
In the main cache, there are 16 cache lines, So 4 bits needed to represent cache index and atleast cache tag should have 6bit width to cache this main memory.



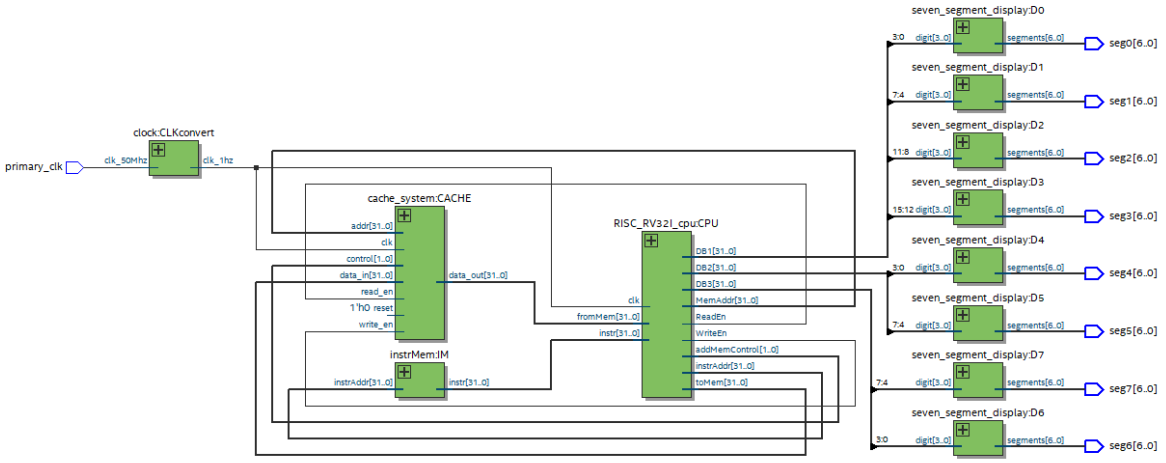
c. Schematics of Data Memory



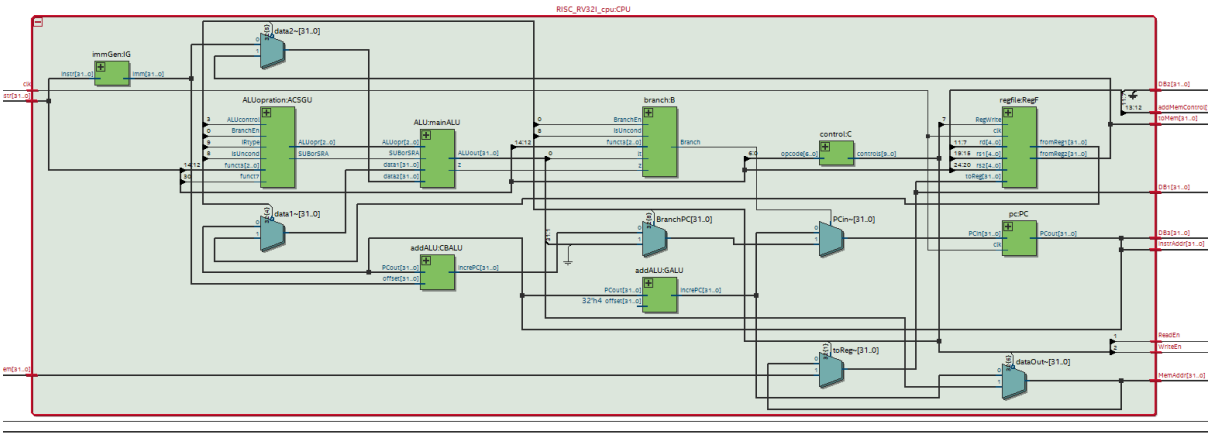
Flow Status

Flow Status	Successful - Sun Feb 12 18:58:45 2023
Quartus Prime Version	17.1.0 Build 590 10/25/2017 SJ Lite Edition
Revision Name	RISC_RV32I_computer_system
Top-level Entity Name	computer_system
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	24,485 / 114,480 (21 %)
Total registers	13304
Total pins	57 / 529 (11 %)
Total virtual pins	0
Total memory bits	2,112 / 3,981,312 (< 1 %)
Embedded Multiplier 9-bit elements	0 / 532 (0 %)
Total PLLs	0 / 4 (0 %)

RTL viewer



Full design



CPU