CPU-1 Documentation

Maciej Mikulski

September 19, 2022

1 Introduction

CPU-1 is my first CPU design. Its goals are improving VHDL coding skills and gaining first experience in designing processors. It contains all the basic features of simple 8-bit CPUs from 70' and 80' and is heavily influended by Z80 design.

2 General architecture and main features

CPU-1 is a simple CPU with 8-bit data bus and 16-bit address bus. It has two working registers and ALU able to perform basic integer operations. CPU-1 is big-endian - high order bytes are stored first (on lower addresses).

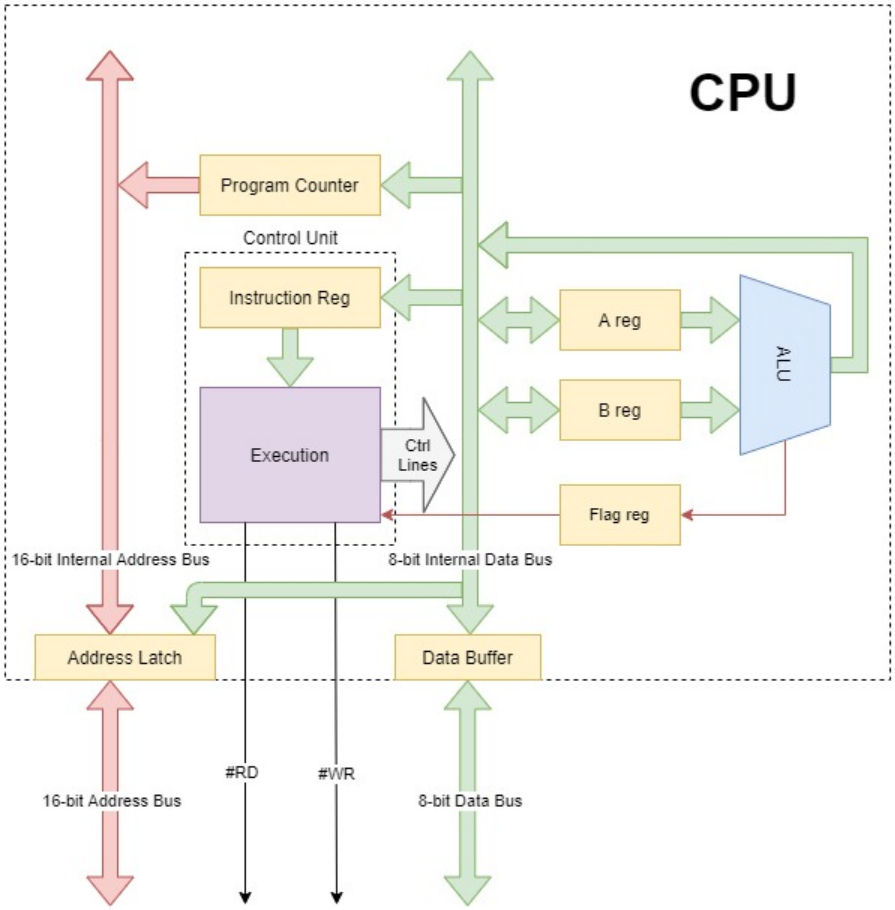

Figure 1: CPU-1 internal block diagram

2.1 ALU and registers

The CPU-1 ALU can perform four operations: addition, logical AND, logical OR and compare. Operands for all operations are stored in A and B registers and the result (in ADD, AND and OR operations) is always stored in A register.

A and B registers can be loaded with contents of external memory or an immediate value. Their contents can be written to any memory address. After

performing operation appropriate flags are set in the *Flag Register (FR)*. These flags are: Carry, Zero, Equal. Flags are continuously generated (combinational circuits), but their values are latched into *FR* during execution of arithmetic instructions (exact behaviour is explained in Instruction Set section).

2.2 Execution Unit

Execution Unit (EU) is composed of *Instruction Register (IR)* and a *State Machine (SM)*. Next state of the *SM* is determined based on the contents of *IR*, states of flags and state of Reset input. *SM* generates internal control signals as well as external control signals (RD and WR) for interfacing with memories and IO devices.

2.3 Program Counter

Program Counter (PC) is a presettable 16-bit register holding the address of currently executed instruction. High and low address bytes are loaded separately into temporary storage and can be then latched into the main *PC* register.

PC has an ability to increment its current value.

2.4 Data and Address Bus

CPU-1 has two internal busses: 8-bit *Data Bus* and 16-bit *Address Bus*. *Data Bus* connects all registers and *ALU* output and is also connected to *Data Buffer (DB)* and *Address Latch (AL)*. *Address Bus* connects *PC* and *AL*.

Data Buffer is a simple bidirectional tri-state buffer connecting *Internal and External Data Busses*.

Address Latch is able to hold address on external *Address Bus*. It can be loaded with the contents of *PC* or its low or high byte can be loaded from the *Data Bus*.

3 Modules description

This section documents internal operation and input/output ports of CPU-1 modules.

3.1 Execution unit

Execution unit is responsible for decoding instruction stored in the *Instruction Register* and coordinating operations of other modules. This module is a FSM that has 19 states - microinstructions. Depending on an opcode stored in the *IR* and the values stored in the *Flag register* state machine is progressing though appropriate states. Control signals are the outputs of this module and their values are generated based on current state of the FSM. This state machine is a Moore machine, as all of its inputs change states synchronously.

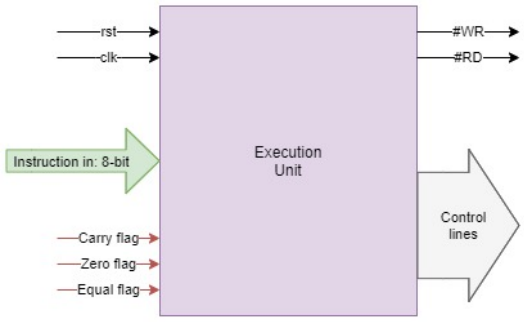

Figure 2: Execution unit block diagram

3.2 Instruction Register

Instruction register holds the opcode of currently executed instruction and constantly outputs it to the *Execution unit*. Its input is connected to the internal data bus.

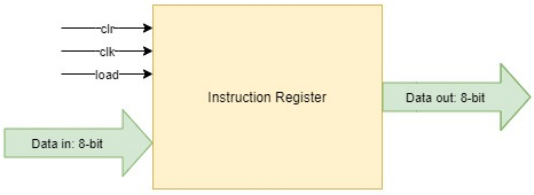

Figure 3: Instruction register block diagram

3.3 ALU

The *Arithmetical logic unit* module has two operand inputs that are connected to the outputs of the A and B registers. Its opsel input is connected to the EU and selects which operation shall be performed. Data output is a tri-state port and is controlled by "en" line. The ALU generates 3 flag signals: Carry, Zero and Equal. Carry and Zero flags are generated continuously and Equal flag is generated while performing Compare operation. Those three signals are connected to the *Flag register*.

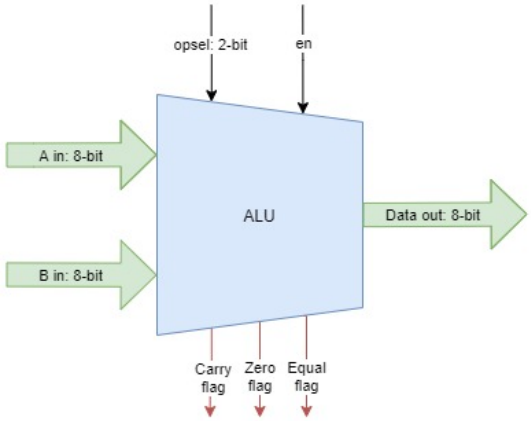

Figure 4: ALU block diagram

3.4 Program Counter

Program counter is the only 16-bit register in the CPU-1. It holds the memory address from which the opcode/operand are currently read. 16-bit output is a tri-state port and is controlled by "en" line. PC has an 8-bit input connected to the internal data bus through which its low and high byte can be loaded. PC has also the ability to increment its contents.

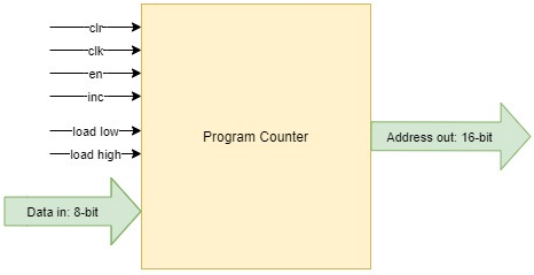

Figure 5: Program counter block diagram

3.5 Working Register

CPU-1 has two 8-bit working registers: *A* and *B*. They are identical in function with the exception that the results of all arithmetic operations are stored int the *A* register. Registers have one 8-bit input that is connected to the internal data bus. Each registers has two outputs - one connected to the *ALU* input and second connected to the internal data bus. On the *ALU* output data stored in the register is always present, while the data bus output is a tri-state port that can be enabled by "en" control line.

Figure 6: Working register block diagram

3.6 Flag register

Flag register can separately latch state of each flag generated by the *ALU*. Its outputs are connetcted to the *Execution unit*.

Figure 7: Flag register block diagram

3.7 Address latch

Address latch is responsible for driving external address bus. It can either pass the output of the *Program counter* or output data latched in its internal register. This register high and low bytes can be loaded from the internal data bus. Output of the *AL* is a tri-state port.

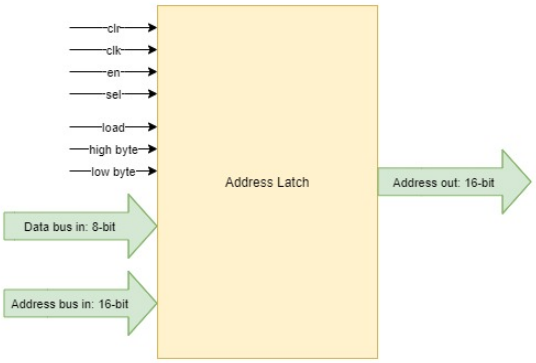

Figure 8: Address latch block diagram

3.8 Data buffer

Data buffer is a bidirectional tri-state port. The direction in which the data flows through the *DB* is controlled by the "dir" control line.

Figure 9: Data buffer block diagram

4 Instruction set

CPU-1 has 14 instructions divided up into 3 groups: arithmetic instructions, load/store instructions and jump instructions.

Mnemonic	OpCode	Description	Operation	Set flags
ADD	0x00	Add A and B	A <- A + B	C, Z
AND	0x01	AND A and B	A <- A & B	Z
OR	0x02	OR A and B	A <- A | B	Z
CP	0x03	Compare A and B	-	EQ
LDA	0x10 imm8	Load A immediate	A <- imm8	-
LDA	0x11 imm16	Load A direct	A <- M(imm16)	-
LDB	0x12 imm8	Load B immediate	B <- imm8	-
LDB	0x13 imm16	Load B direct	B <- M(imm16)	-
STA	0x14 imm16	Store A direct	M(imm16) <- A	-
STB	0x15 imm16	Store B direct	M(imm16) <- B	-
JMP	0x20 imm16	Jump to immediate	PC <- imm16	-
JNZ	0x21 imm16	Jump to immediate if not zero	PC <- imm16 if Z=0	-
JPC	0x22 imm16	Jump to immediate if carry	PC <- imm16 if C=1	-
JEQ	0x23 imm16	Jump to immediate if euqal	PC <- imm16 if EQ=1	-

Table 1: CPU-1 instruction set

Notation	Description
imm8	8-bit unsigned integer value
imm16	16-bit unsigned integer value
M(N)	contents of memory at address N
-	none

Table 2: Instruction notation summary

4.1 Addressing modes

CPU-1 has 3 addressing modes.

4.1.1 Implied addressing

This addressing mode is used in all arithmetic instructions. Implied operands are *A* and *B* registers and the destination of ADD, AND and OR operations is the *A* register.

4.1.2 Immediate addressing

Single byte immediate addressing is used in Load A and Load B instructions. Byte following the opcode in memory is loaded into specified register.

Dual byte immediate addressing is used in all Jump instructions. If a jump is executed two bytes following the opcode are loaded into the *Program Counter*. Higher order byte is stored first in the memory, as CPU-1 is a big-endian machine.

4.1.3 Direct addressing

This addressing mode is used in load and store instructions. In load instructions A or B register is loaded with memory contents from the address specified by two bytes following load instruction opcode. In store instructions, contents of registers are written into memory location specified by two bytes following the instruction opcode.

4.2 Instructions description

All instructions are divided-up into T-states. One T-state lasts for one clock cycle and begins on its rising edge. Each timing diagram shows one T-state before and after instruction's T-states. Those additional states are called "Tx" and are added in order to improve of appearance of the generated waveforms. Each diagram shows states of all control signals belonging to CPU modules active during execution of particular instruction.

All loads into registers happen in the middle of T-cycle - on the falling edge of the system clock in order to provide sufficient set-up time on data and address busses and on control lines.

4.2.1 Arithmetic instructions

All arithmetic instructions are executed in two clock cycles. T1 state is *Instruction Fetch* state in which data from memory is loaded into *IR*. During T2, ALU output is enabled and data is latched into A register in ADD, AND and OR instructions. Flag states are latched into the *FR*. *PC* is incremented.

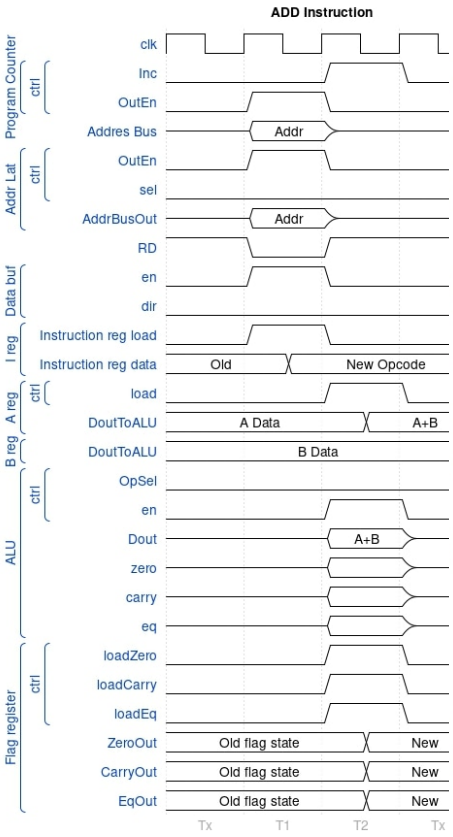

Figure 10: ADD instruction waveform

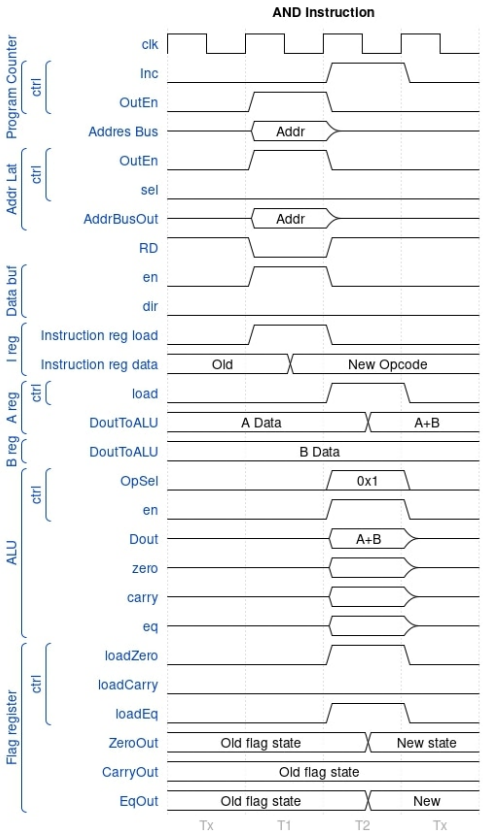

Figure 11: AND instruction waveform

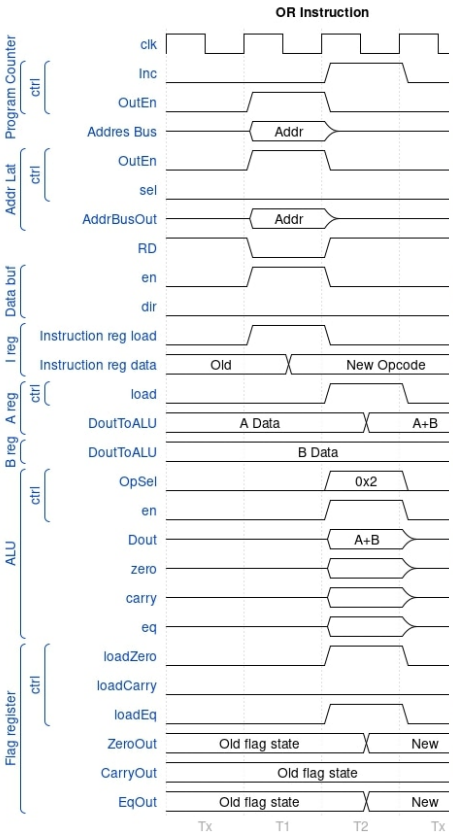

Figure 12: OR instruction waveform

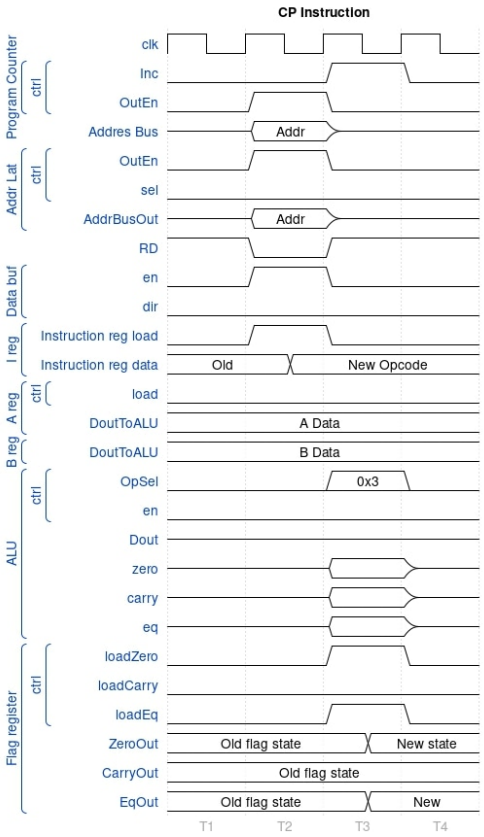

Figure 13: CP instruction waveform

4.2.2 Load register immediate

LDA imm8 and LDB imm8 instructions are executed in 3 T-states. During T1 state data from memory is loaded into *IR*. During T2 *PC* is incremented. In T3 state imm8 byte is loaded from memory into respective register.

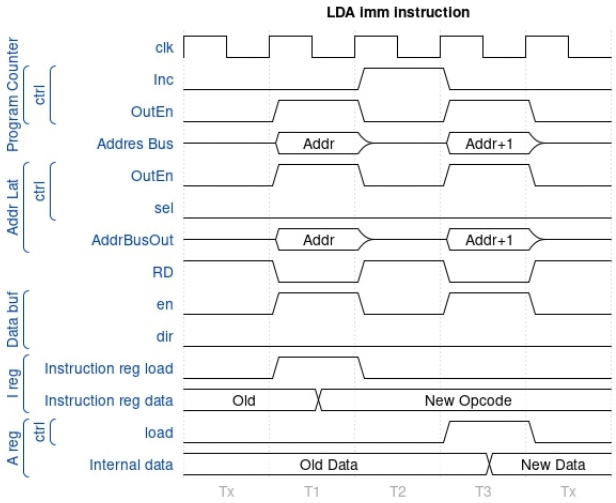

Figure 14: LDA imm8 instruction waveform

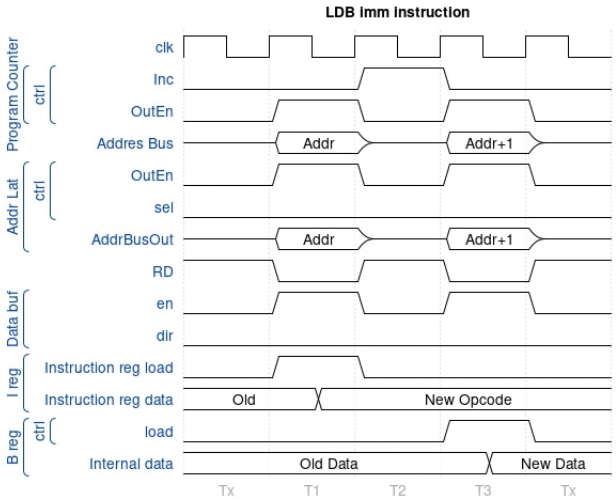

Figure 15: LDB imm8 instruction waveform

4.2.3 Load/Store register direct

LDA (imm16), LDB (imm16), STA (imm16) and STB (imm16) instructions are executed in 7 T-states. T-states T1 through T6 are identical in all of the above instructions. During T1 state data from memory is loaded into *IR*. During T2 *PC* is incremented. In T3 high byte of imm16 memory address is loaded into *AL*. In T4, *PC* is incremented. In T5 low byte of imm16 memory address is loaded into *AL*. In T6 *PC* is incremented. In load instructions during T7 state, contents of the memory location under imm16 address are stored in respective register. In store instructions, contents of the respective register are stored in memory under the imm16 address.

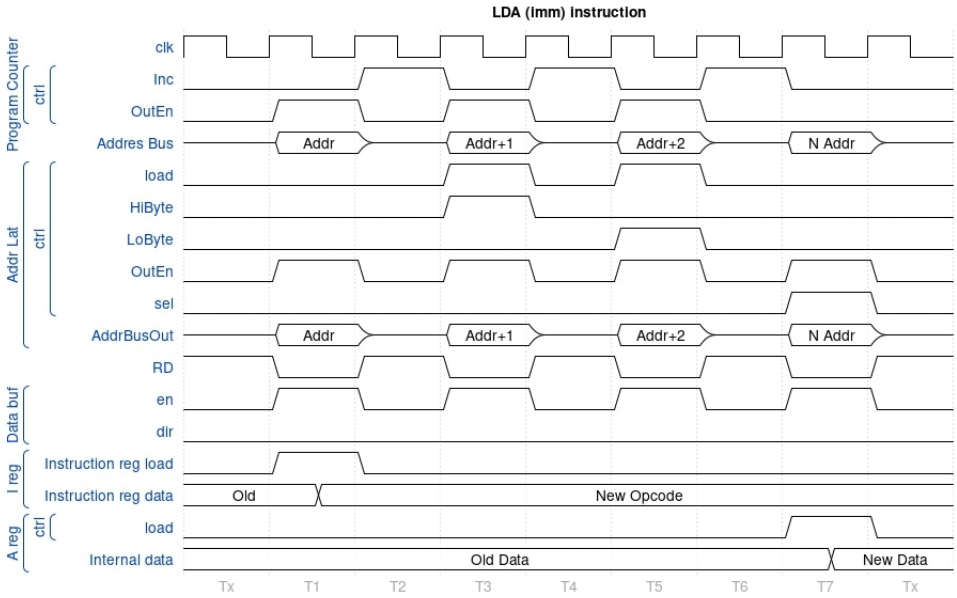

Figure 16: LDA (imm16) instruction waveform

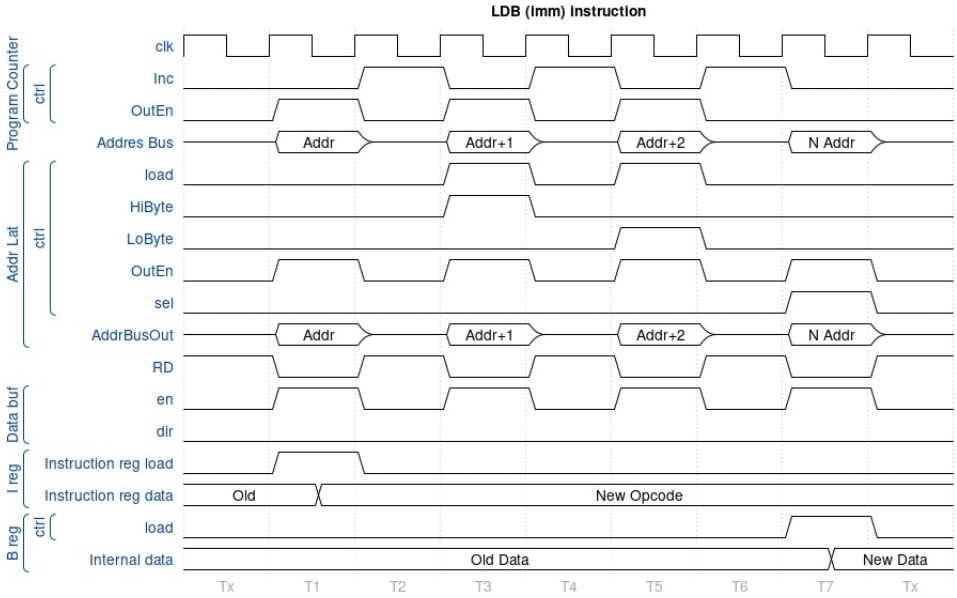

Figure 17: LDB (imm16) instruction waveform

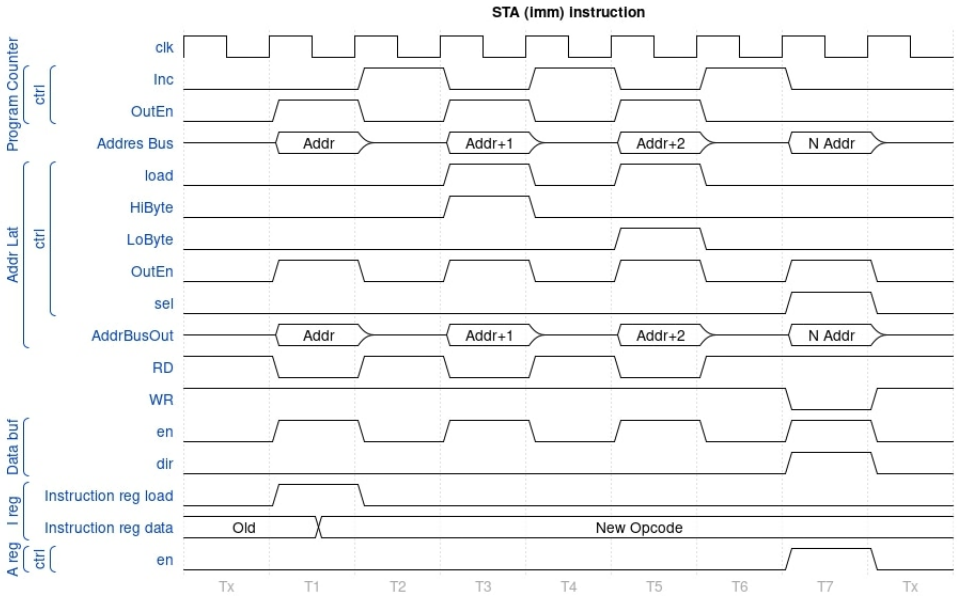

Figure 18: STA (imm16) instruction waveform

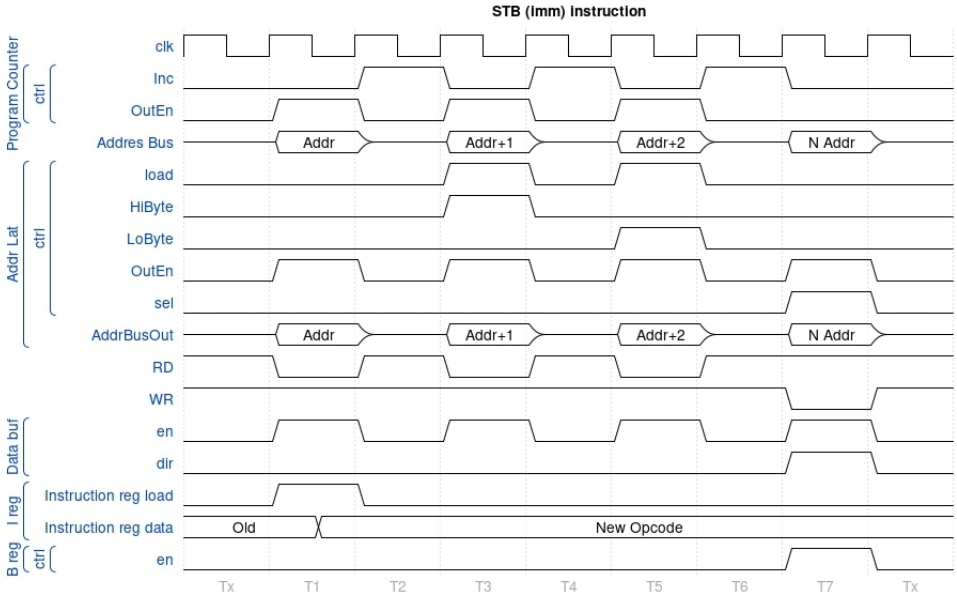

Figure 19: STB (imm16) instruction waveform

4.2.4 Jump instructions

All jump instructions have the same structure with the exception that in JNZ, JPC and JEQ instructions T-states from T3 to T6 are executed conditionally. T1 state is *Instruction Fetch* state in which data from memory is loaded into *IR*. During T2 the *PC* is incremented and in the case of conditional jump instructions state of respective flag is checked. In T3 high byte of new *PC* value is loaded. In T4, *PC* is incremented. In T5 low byte of new *PC* value is loaded. In T6 new value of *PC* is latched from temporary storage into main *PC* register.

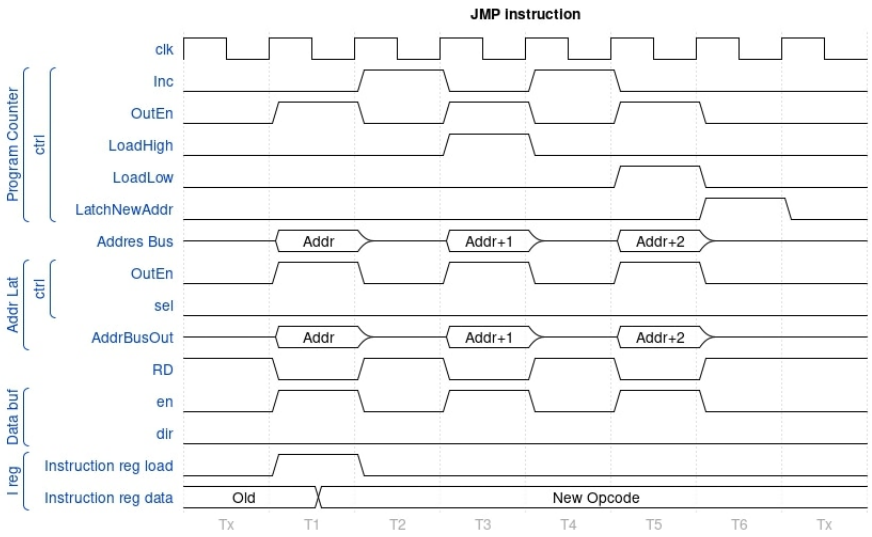

Figure 20: JMP instruction waveform

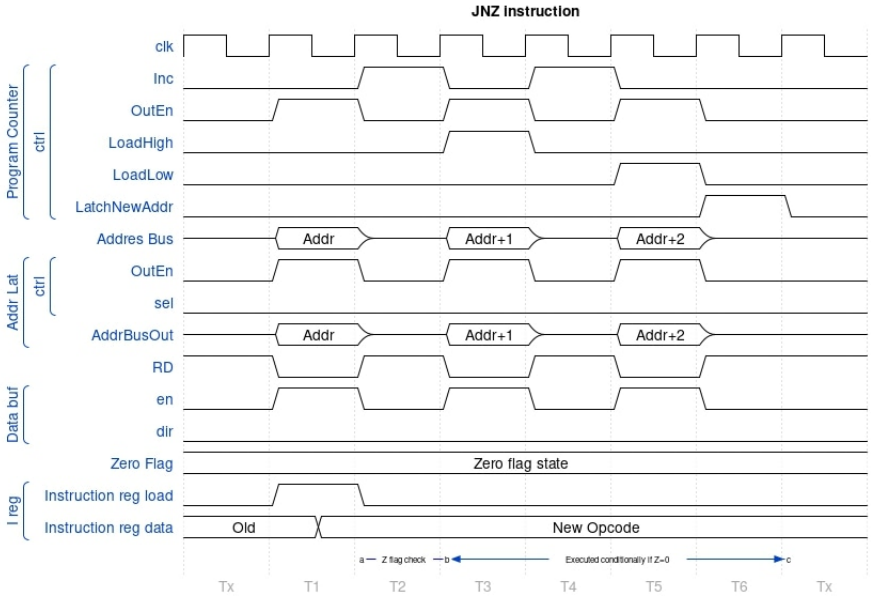

Figure 21: JNZ instruction waveform

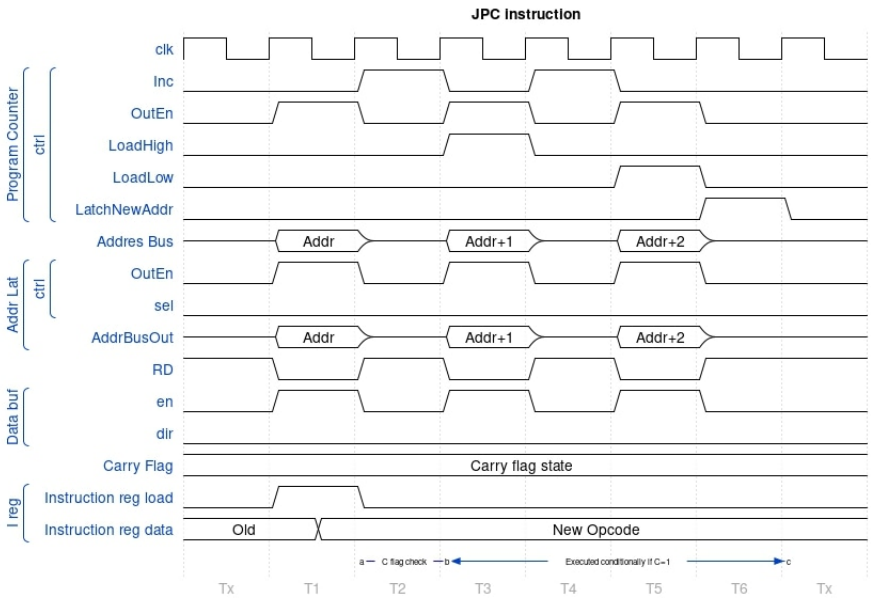

Figure 22: JPC instruction waveform

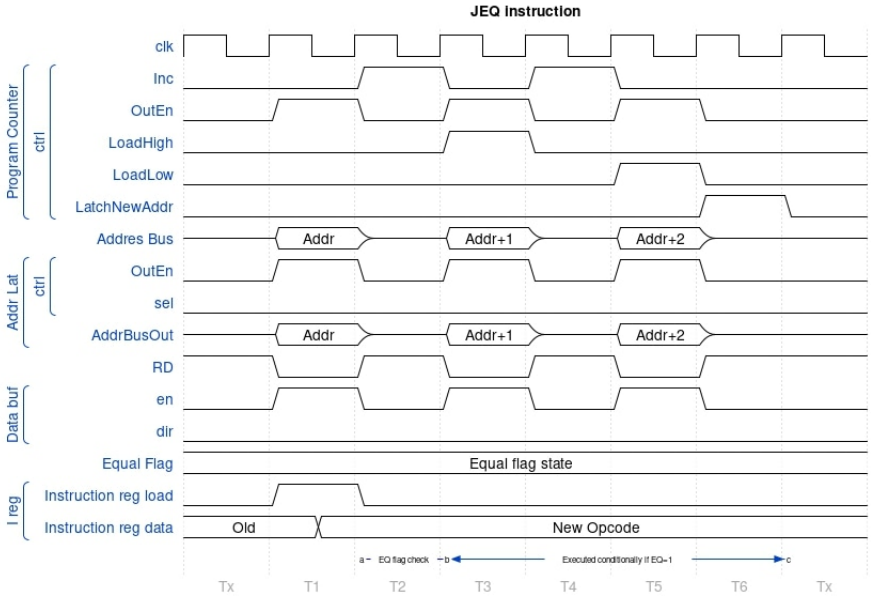

Figure 23: JEQ instruction waveform

5 Basic CPU-1 system

6 Programming

7 Conclusion