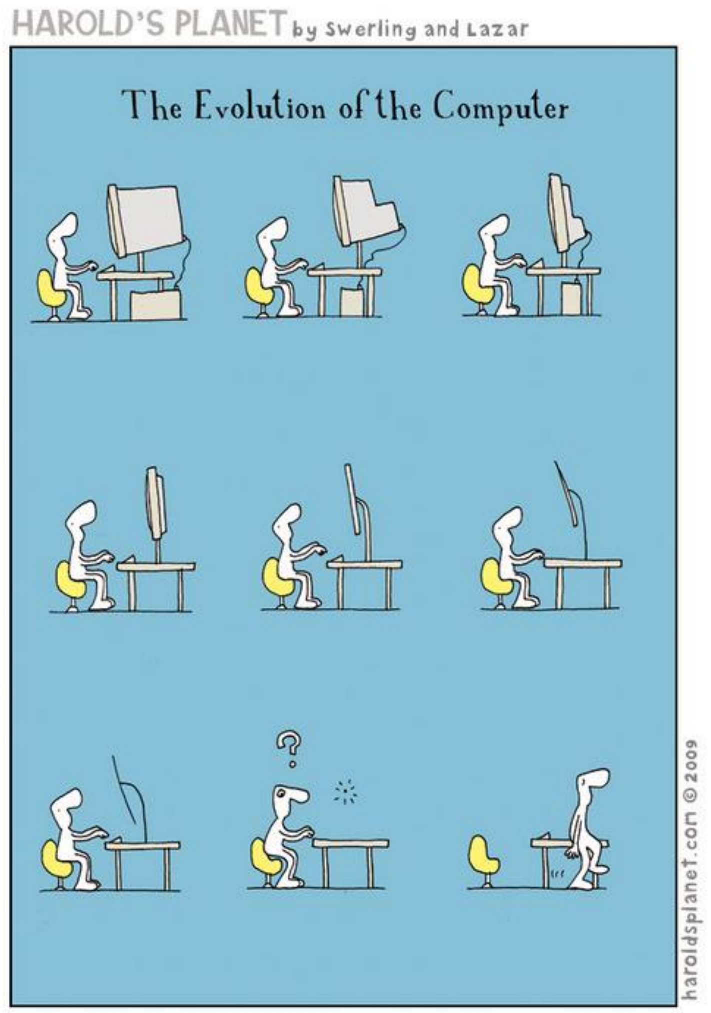
MANAGEMENT AND ANALYSIS OF PHYSICS DATASET (MOD. A)

SAP-1 computer

Simple As Possible computer

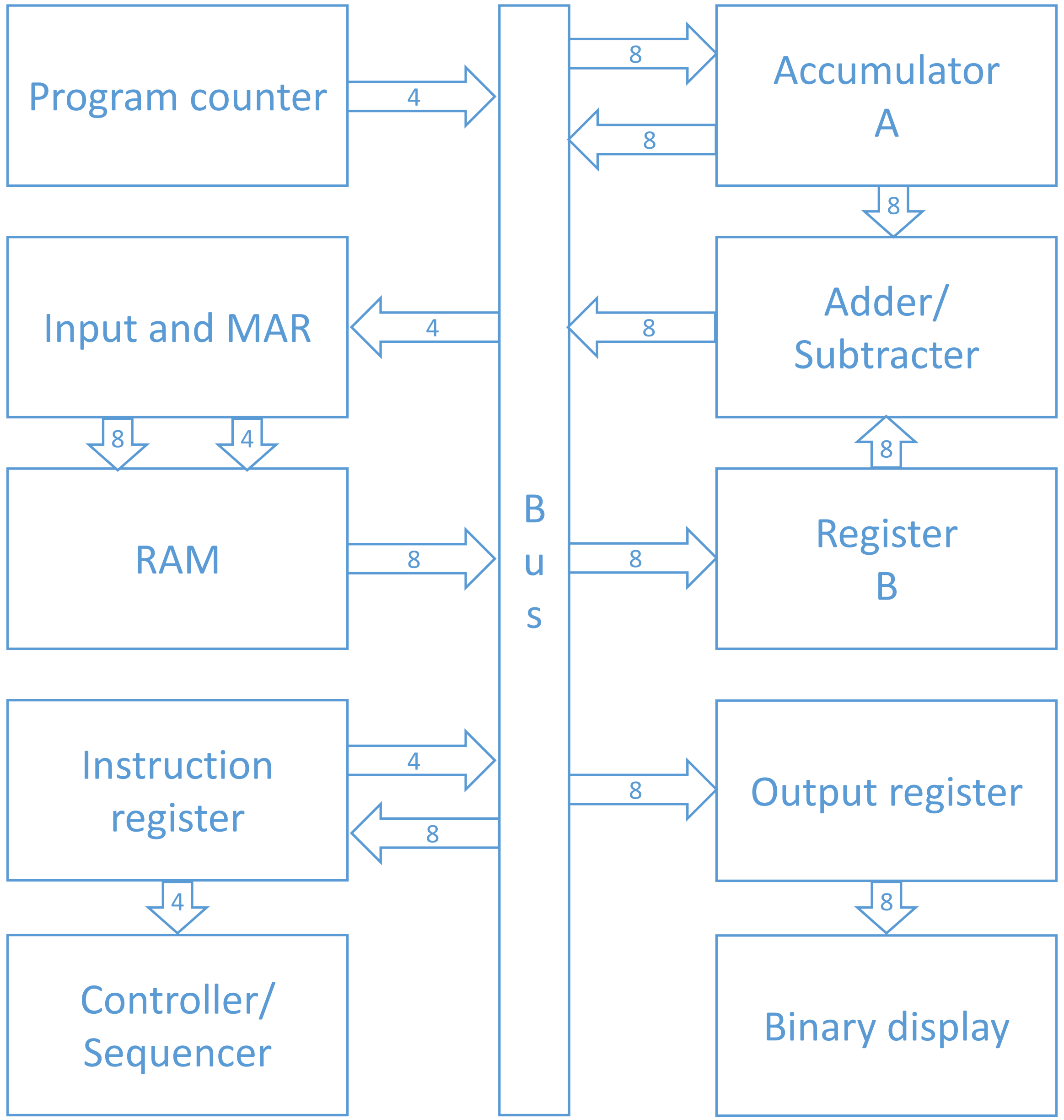
- A computer designed to show all the main ideas behind computer operation without unnecessary detail
- As simple as possible, but no simpler
- Three different generation will be presented SAP-1-2-3
- It is the evolution towards modern computer


HAROLD'S PLANET by Swerling and Lazar
The Evolution of the Computer
haroldsplanet.com © 2009

SAP-1

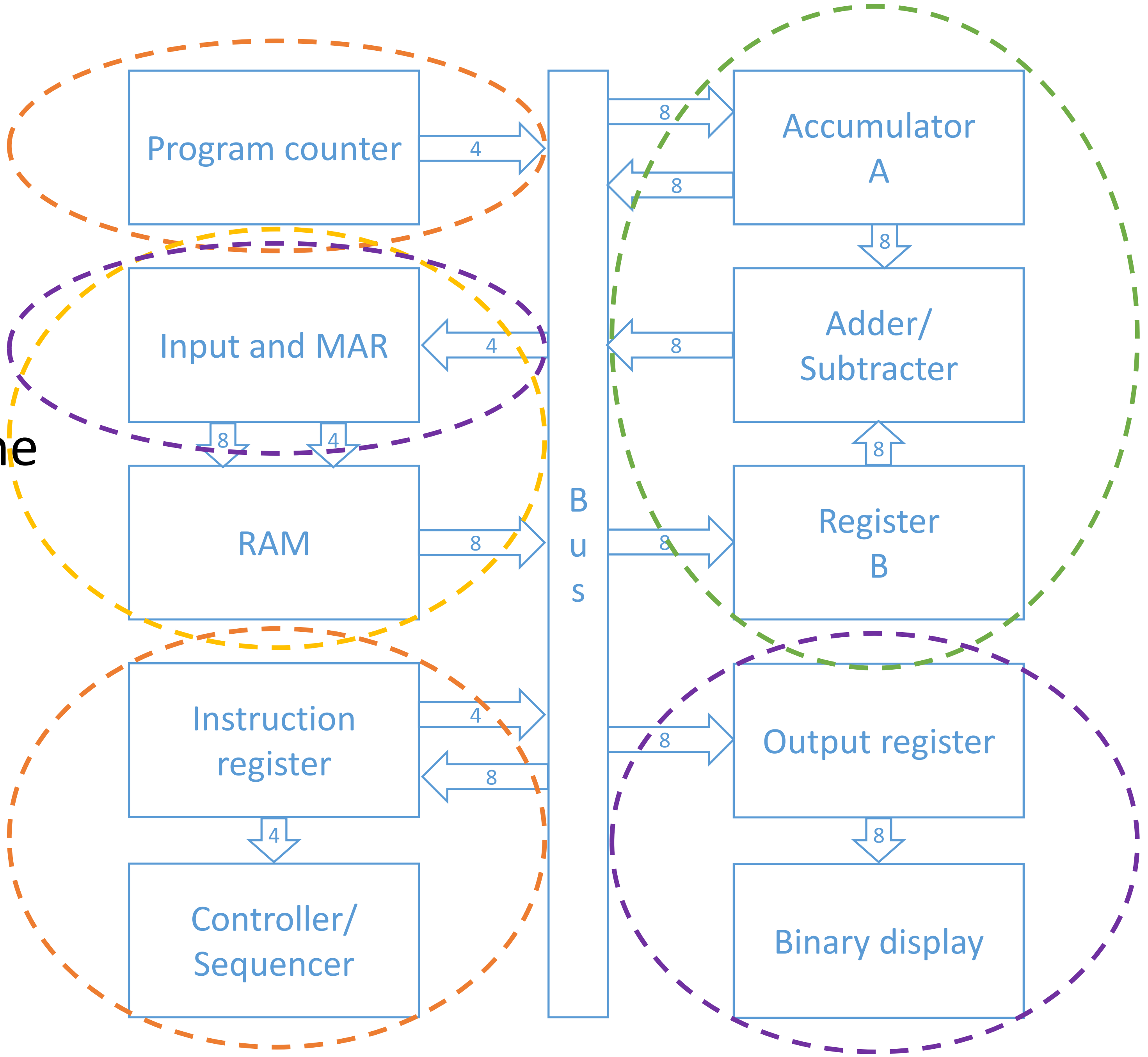
Architecture

- Simplified schema without clock, reset and controls
- All the register outputs to the bus are three-state



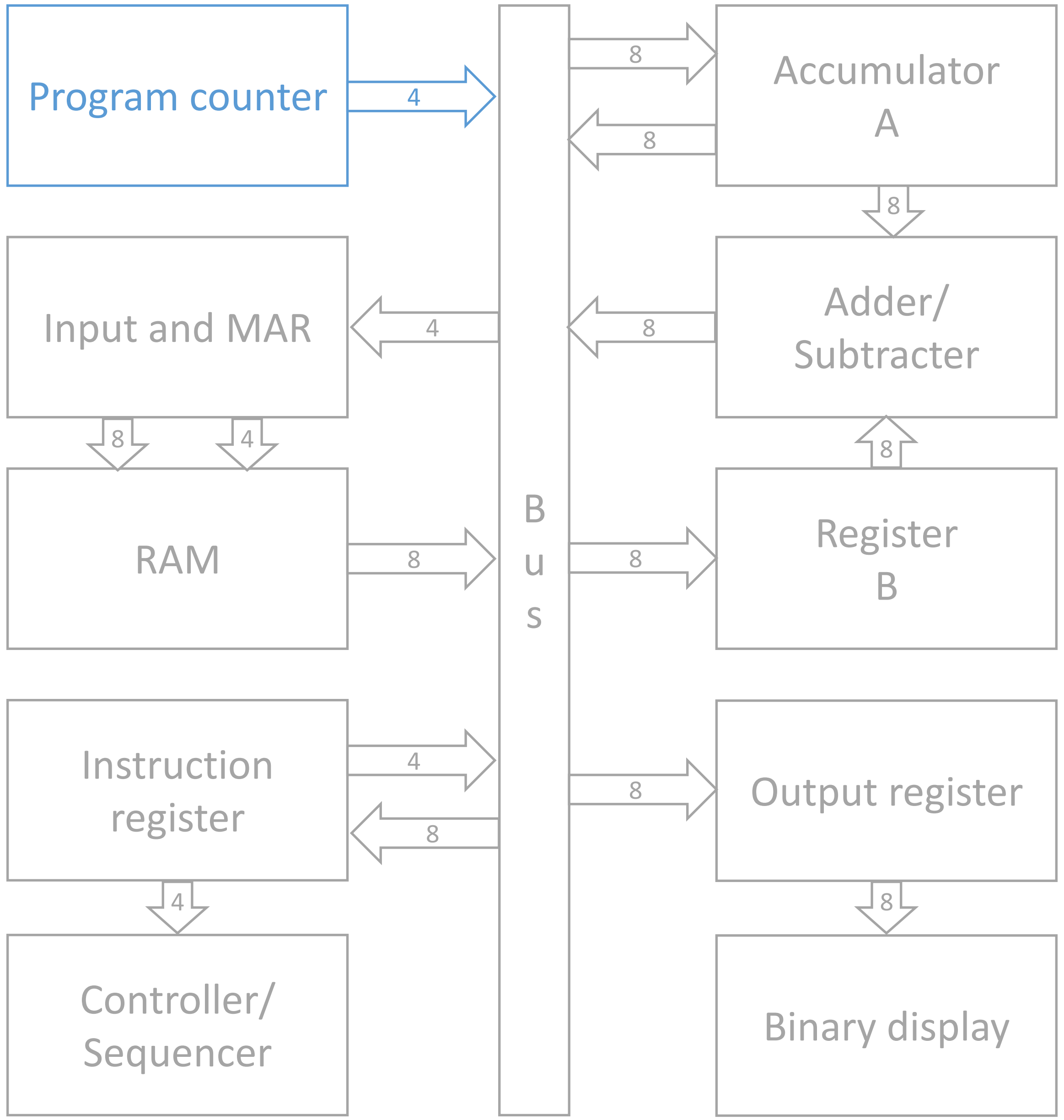
Architecture

- Simplified schema without clock, reset and controls
- All the register outputs to the bus are three-state
- It is built by a Control Unit, an ALU, a Memory and an I/O Unit



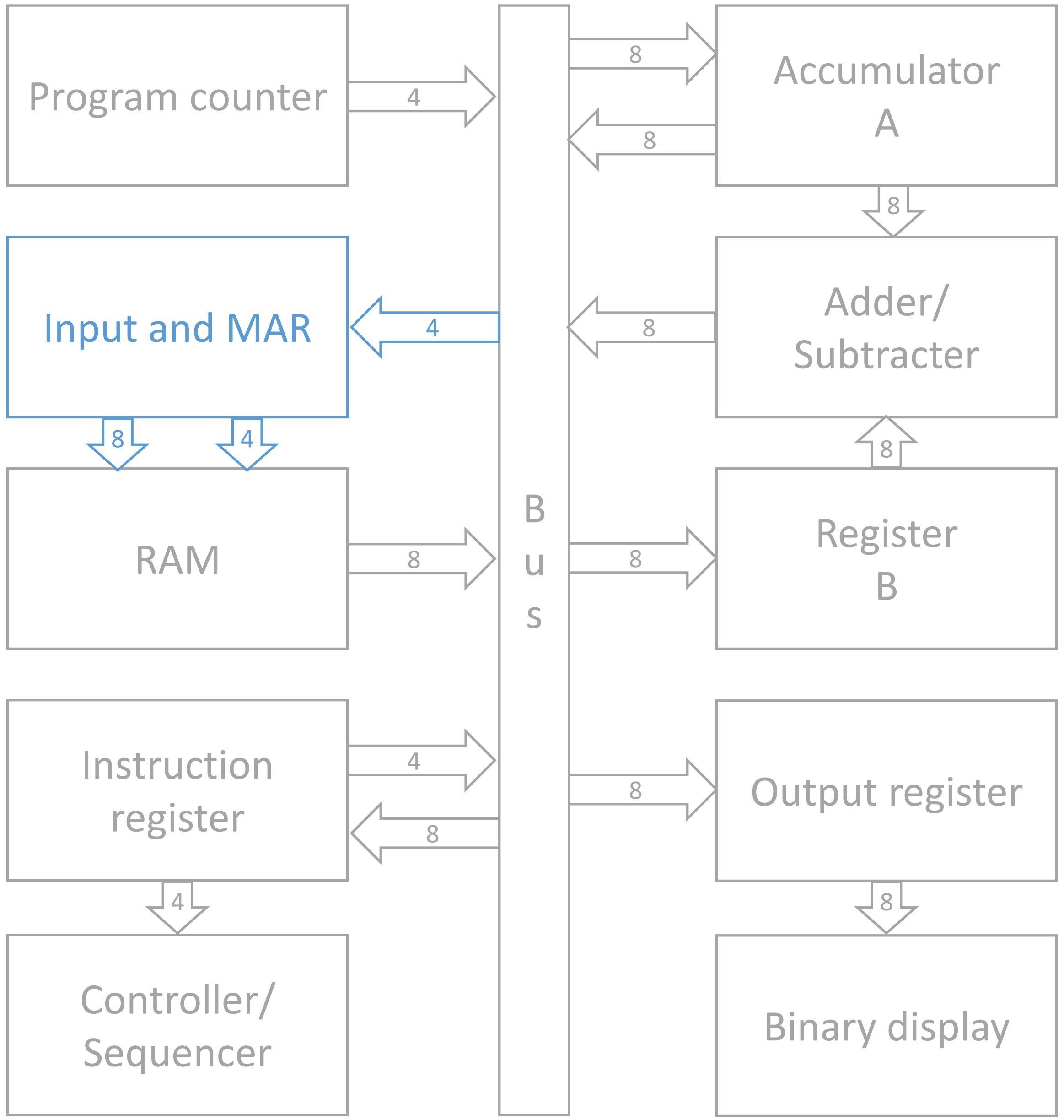
Architecture

- Program counter counts from 0000 to 1111
- It sends to the memory the address of the next instruction to be fetch and executed
- Before a run it is reset to 0000
- It is incremented each fetch and execution cycle
- It is also called pointer because it points to the address where the next instruction is stored



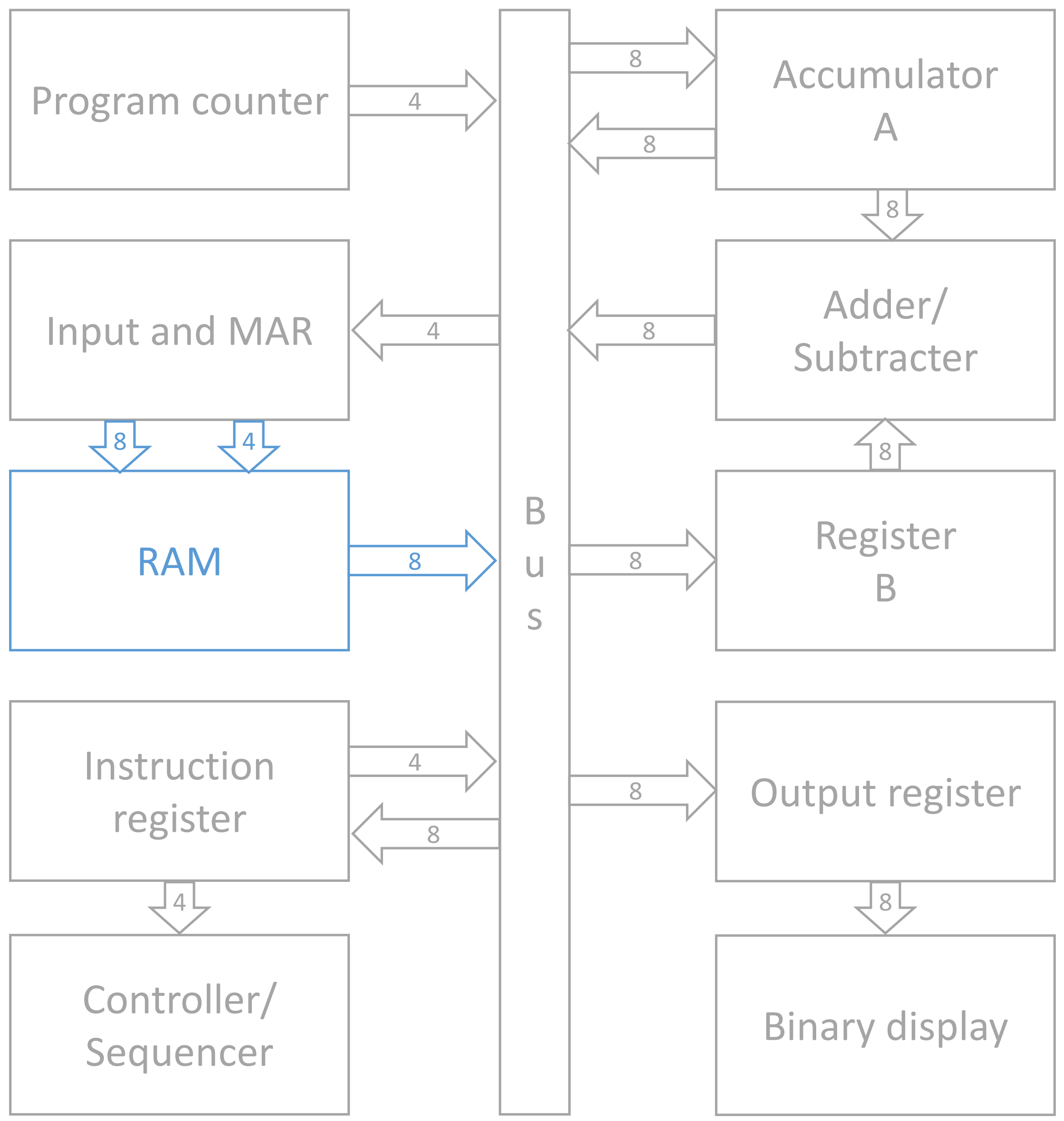
Architecture

- Input and MAR includes
    - Two switch registers (input registers) for address and data
    - A Memory Address Register (MAR)
- The input switch registers are used to write into the RAM the instruction and data words before a run
- MAR is used to latch the program counter address and to apply this address to the RAM



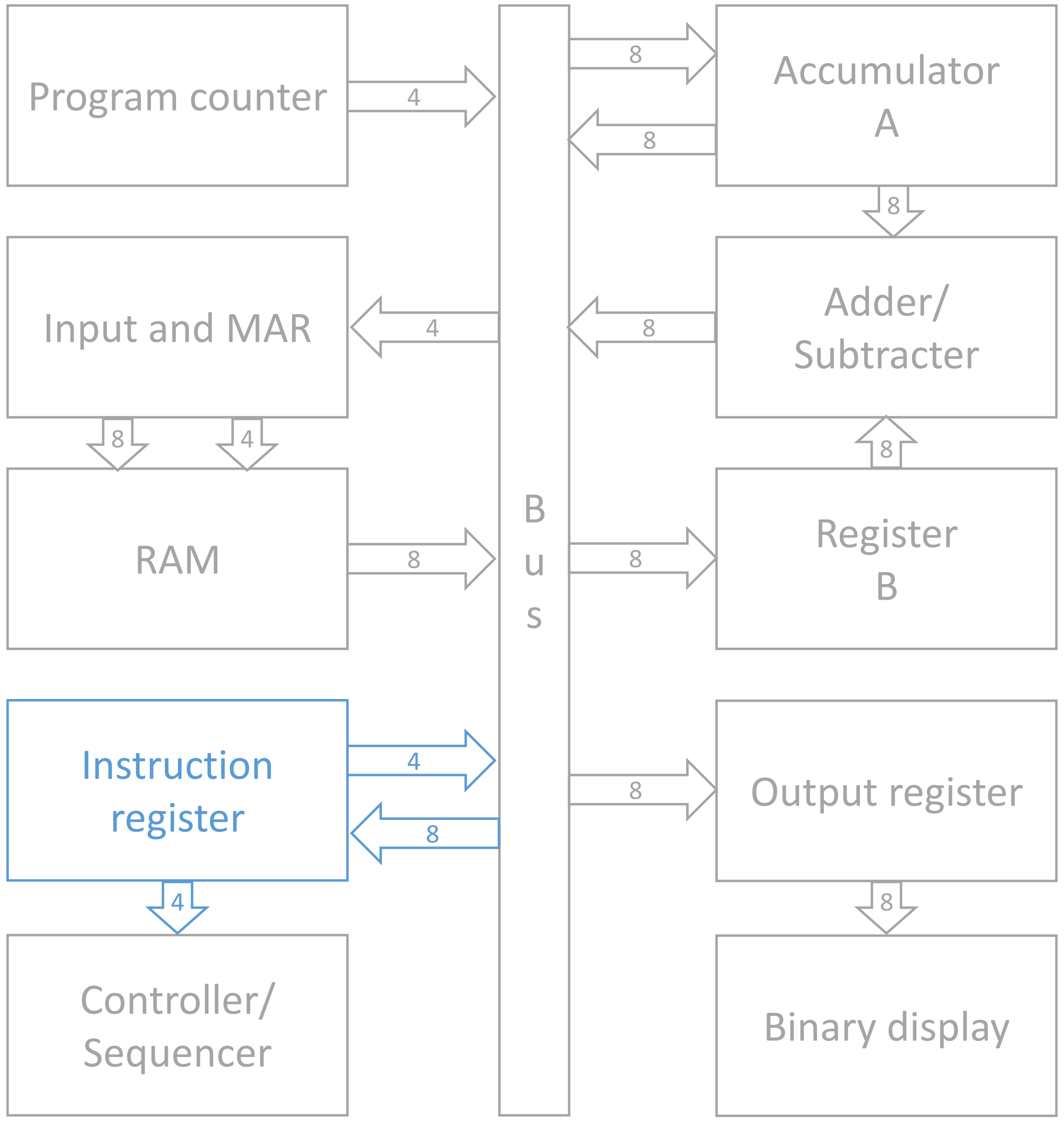
Architecture

- The memory block is a Random Access Memory (RAM) organized in 16x8-bit
- During a run RAM receive addresses from MAR and perform a read operation
- The instruction or data stored in RAM is placed in the bus to be used by some other block



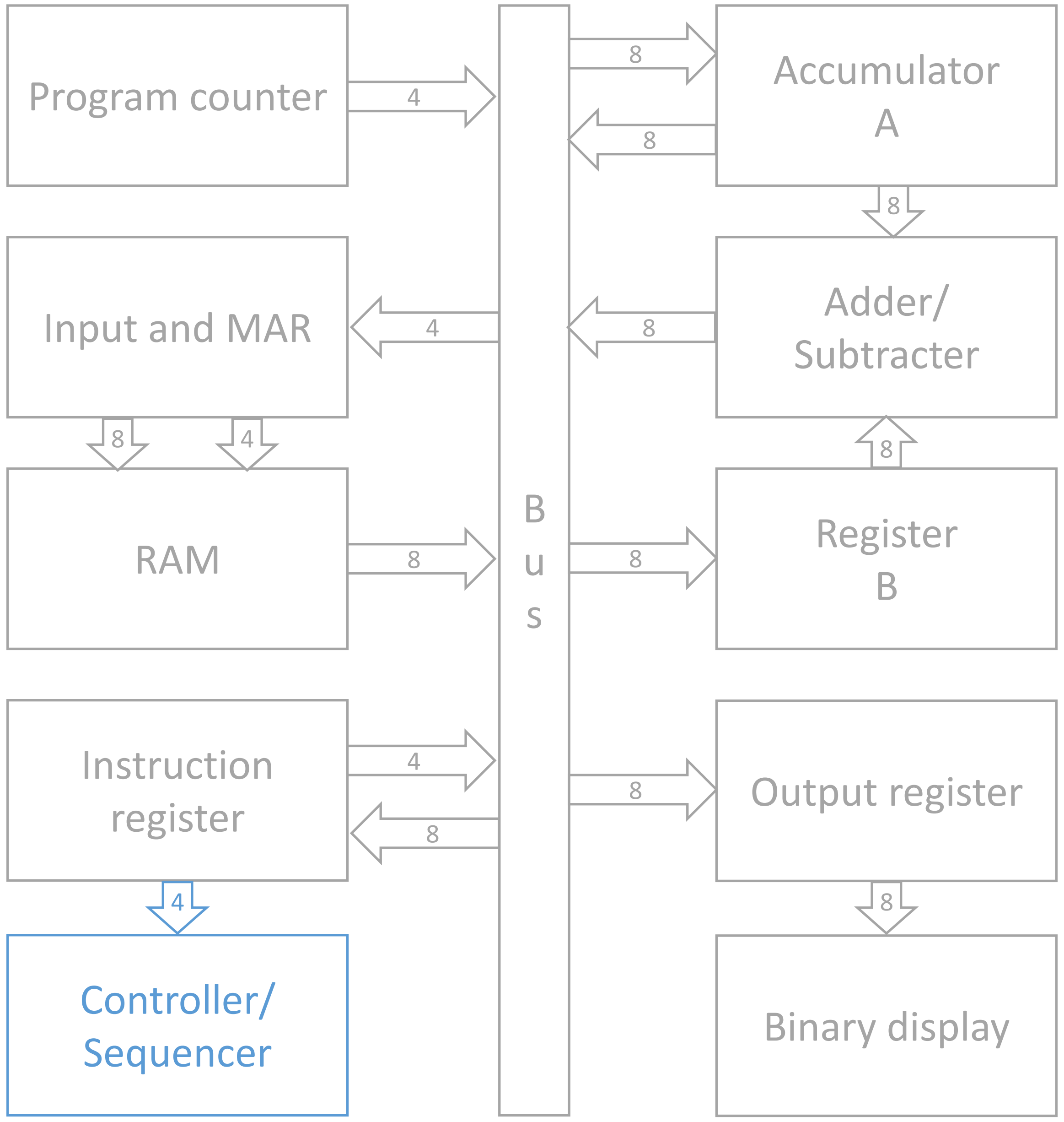
Architecture

- After reading an instruction from the memory, its content is placed on a register
- The content is split into two parts, one for the controller/sequencer and the other it is available on the bus



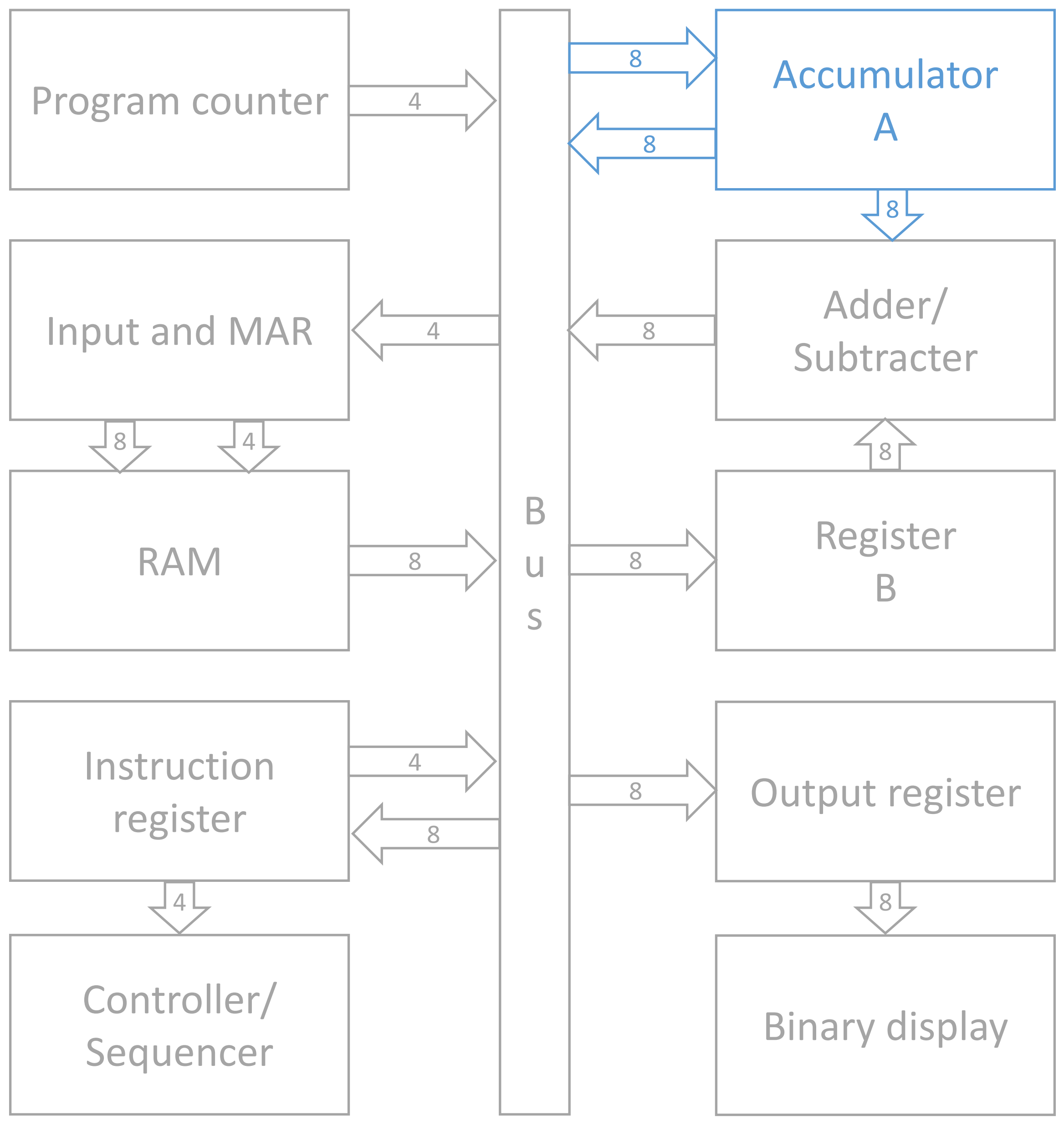
Architecture

- Controller/sequencer distribute the reset signal to the program counter and the instruction register at the beginning of a run
- It synchronize all the blocks providing the clock signal
- It provides a 12-bit word that is used for controlling all the blocks



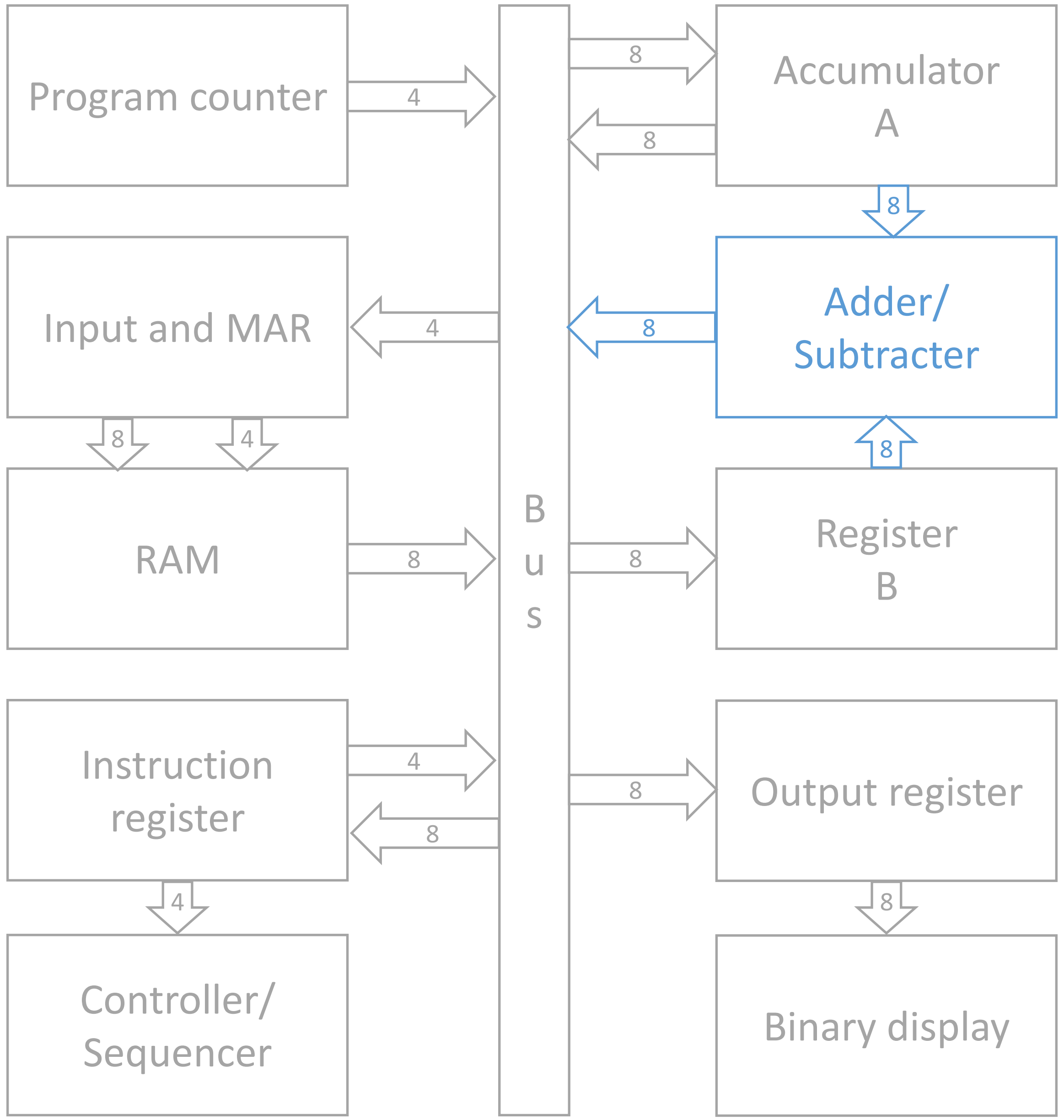
Architecture

- During a run intermediate answers are stored in the Accumulator
- It is a register that is placed on the bus and that have also a direct access to the adder/subtracter



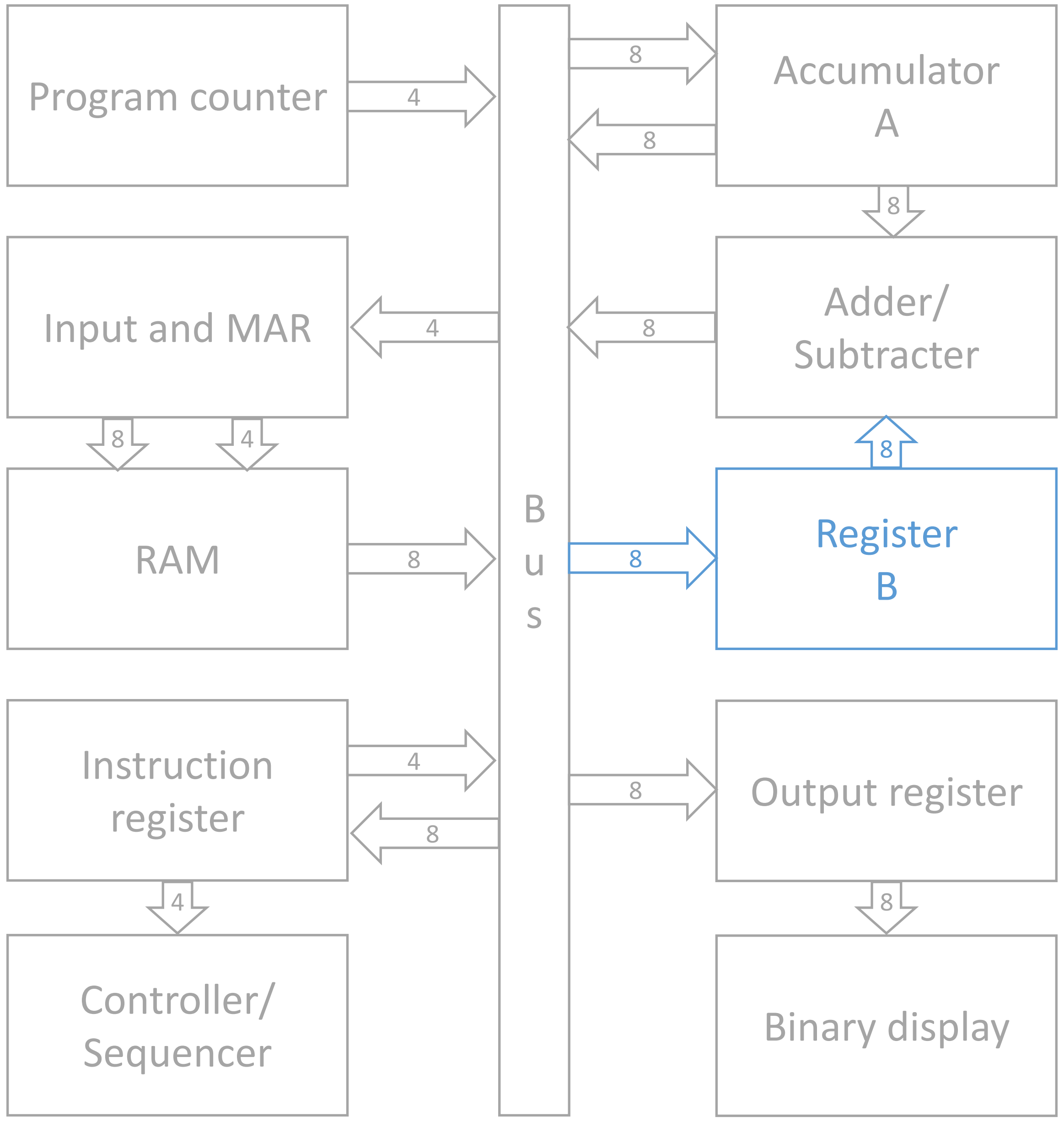
Architecture

- 2’s complement adder/subtracter
- $S = A + B$ or $S = A + B'$
- It is asynchronous



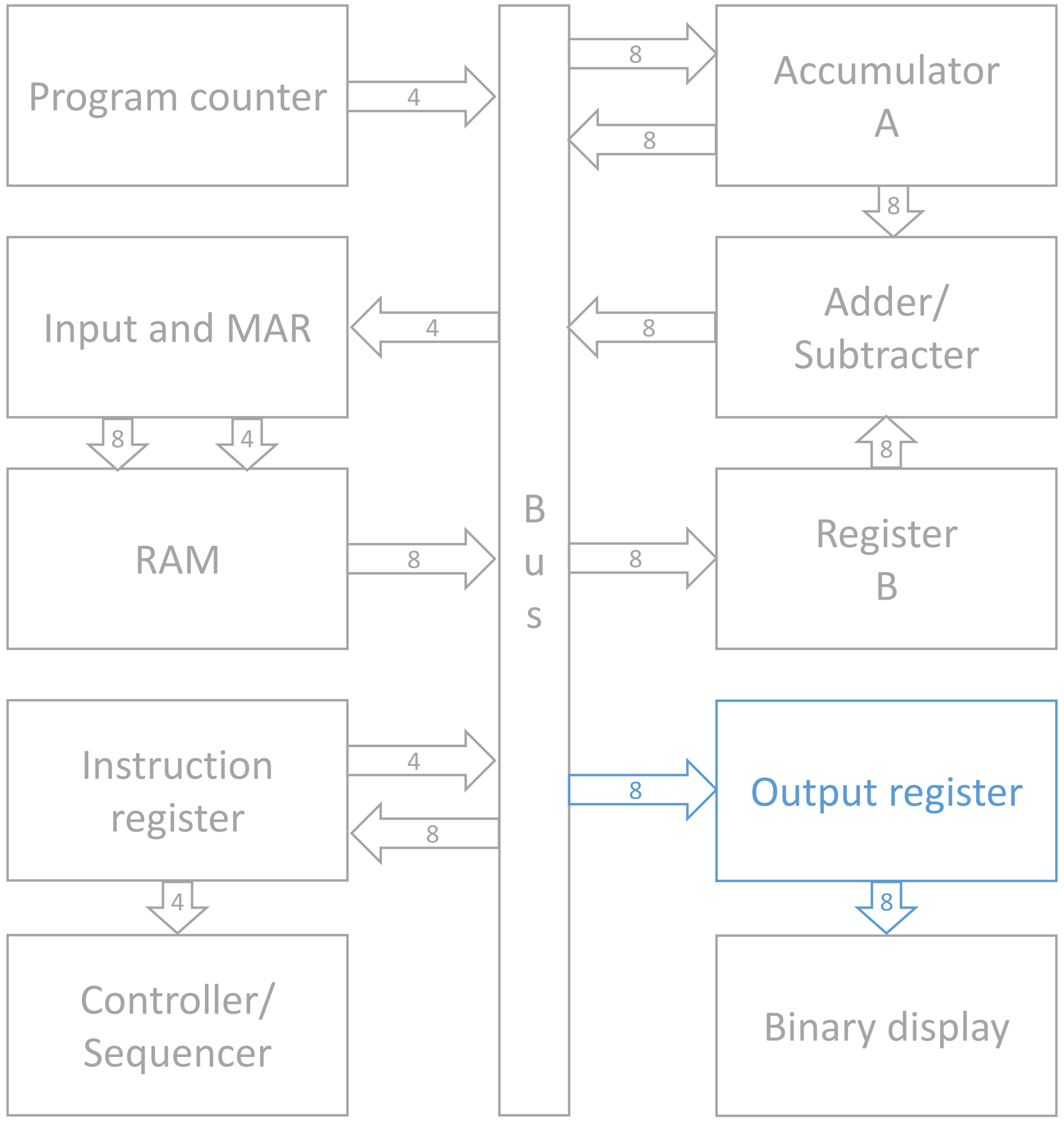
Architecture

- It is a register like the accumulator
- But it cannot be read from the bus



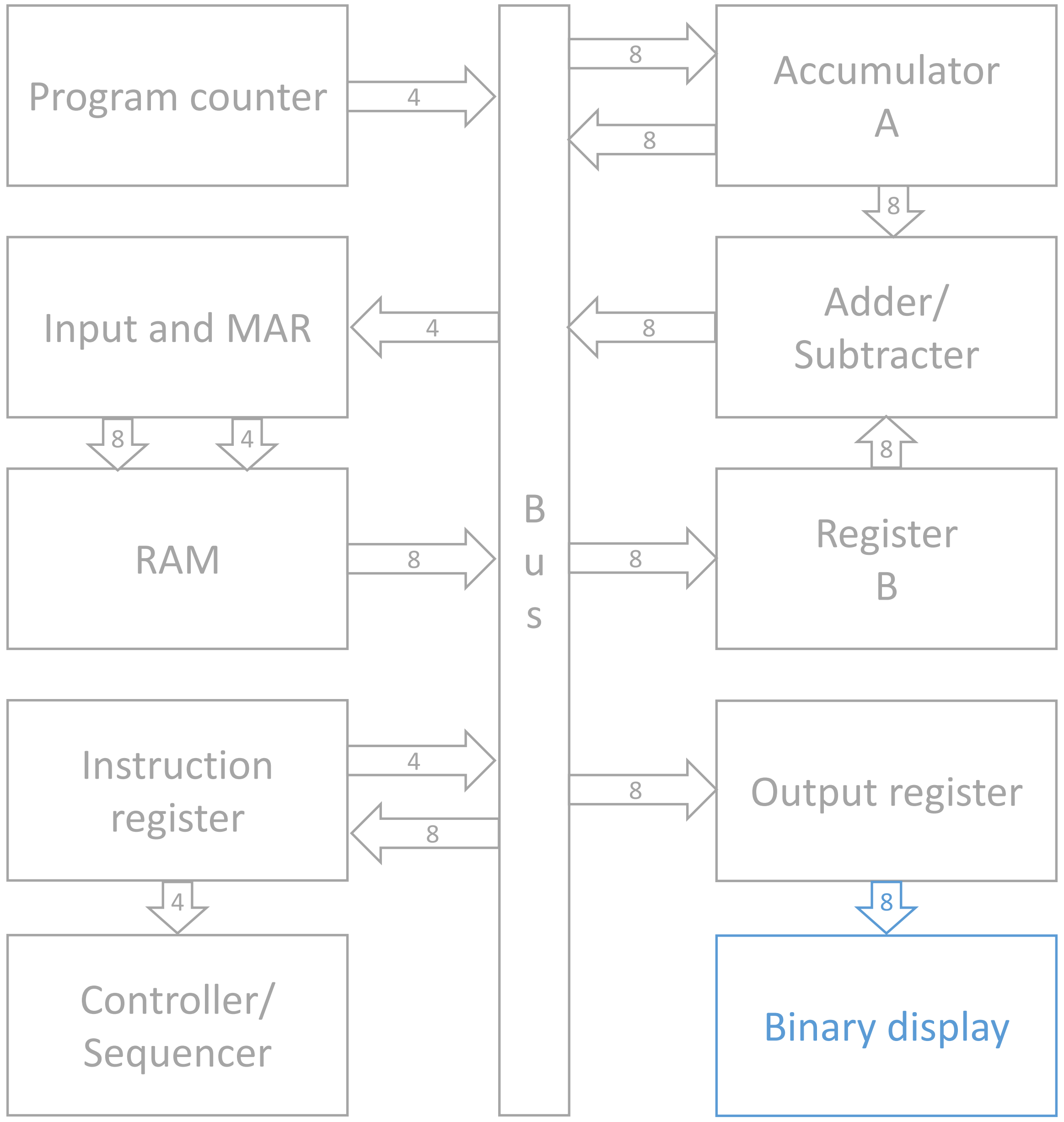
Architecture

- It is a register like B
- At the end of a run the accumulator contents is transferred to the output register
- It is used as a port to the outside word



Architecture

- The outcome of a run is provided to an interface that drives peripheral devices
- In SAP-1 a binary display is used to show the content of the output register
- It is simply built with 8 LEDs



Instruction set

- A computer needs to be programmed (damn it!)
- A program is a series of instructions that have to be loaded into memory before running
- For programming we need to know the instruction set
- SAP-1 has 5 instructions: LDA, ADD, SUB, OUT and HLT
- Abbreviated instructions are called mnemonics

LDA

- LoaD the Accumulator
- It loads the accumulator with the content of a memory address
- It has to include the memory address to be loaded to the accumulator
- For instance, “LDA 0x8” means to copy the content of memory at address 0x8 to the accumulator register

$$M_8 = 00000011 \xrightarrow{\text{LDA 0x8}} A = 00000011$$

ADD

- It add the content of the of a memory address to the accumulator content
- It has to include the memory address to be added to the accumulator
- For instance, “ADD 0x9” means to copy the content of memory at address 0x9 to the register B and to load the output of the adder to the accumulator register

$$M_9 = 00000011 \quad A = 00000100 \xrightarrow{\text{ADD 0x9}} A = 00000111$$

SUB

- It subtract the content of the of a memory address from the accumulator content
- It has to include the memory address to be subtracted from the accumulator
- For instance, “SUB 0xC” means to copy the content of memory at address 0xC to the register B and to load the output of the subtracter to the accumulator register

$$M_C = 00000011 \quad A = 00000111 \xrightarrow{\text{SUB 0xC}} A = 00000100$$

OUT

- It copy the accumulator content to the output port
- It does not include any memory address

$$A = 00000100 \xrightarrow{\text{OUT}} OUT = 00000100$$

HLT

- It stands for HaLT
- It marks the end of the run
- It does not include any memory address
- It is mandatory to include it at the end of any program otherwise there would be meaningless answers caused by runaway processing

Operation code

- To encode the instructions in binary format we need the so called op code
- The op code is mapped in the upper nibble of the instruction while the lower is the operand
- An 8-bit instruction is split into two nibbles: op code and operand

Mnemonics	Op Code
LDA	0000
ADD	0001
SUB	0010
OUT	1110
HLT	1111

- A program written with mnemonics is expressed in Assembly language (source program)
- A program written with op code and operand is expressed in Machine language (object program)
- In SAP-1 the operator translates the source into object program when programming the switch register

Source program example

Address	Data
0x0	LDA 0x8
0x1	ADD 0x9
0x2	ADD 0xA
0x3	SUB 0xB
0x4	OUT
0x5	HLT
0x6	0xFF
0x7	0xFF
0x8	0x4
0x9	0x6
0xA	0x1
0xB	0x2
0xC	0xFF
0xD	0xFF
0xE	0xFF
0xF	0xFF

$$M_8 = 00000100 \xrightarrow{\text{LDA 0x8}} A = 00000100$$

$$M_9 = 00000110 \quad A = 00000100 \xrightarrow{\text{ADD 0x9}} A = 00001010$$

$$M_{10} = 00000001 \quad A = 00001010 \xrightarrow{\text{ADD 0xA}} A = 00001011$$

$$M_{11} = 00000010 \quad A = 00001011 \xrightarrow{\text{SUB 0xB}} A = 00001001$$

$$A = 00001001 \xrightarrow{\text{OUT}} OUT = 00001001$$

$$\text{HLT}$$

Object program example

Address	Data
0x0	LDA 0x8
0x1	ADD 0x9
0x2	ADD 0xA
0x3	SUB 0xB
0x4	OUT
0x5	HLT
0x6	0xFF
0x7	0xFF
0x8	0x4
0x9	0x6
0xA	0x1
0xB	0x2
0xC	0xFF
0xD	0xFF
0xE	0xFF
0xF	0xFF

$\Rightarrow$

Address	Data
0x0	0x08
0x1	0x19
0x2	0x1A
0x3	0x2B
0x4	0xEX
0x5	0xFX
0x6	0xXX
0x7	0xXX
0x8	0x04
0x9	0x06
0xA	0x01
0xB	0x02
0xC	0xXX
0xD	0xXX
0xE	0xXX
0xF	0xXX

Timing states

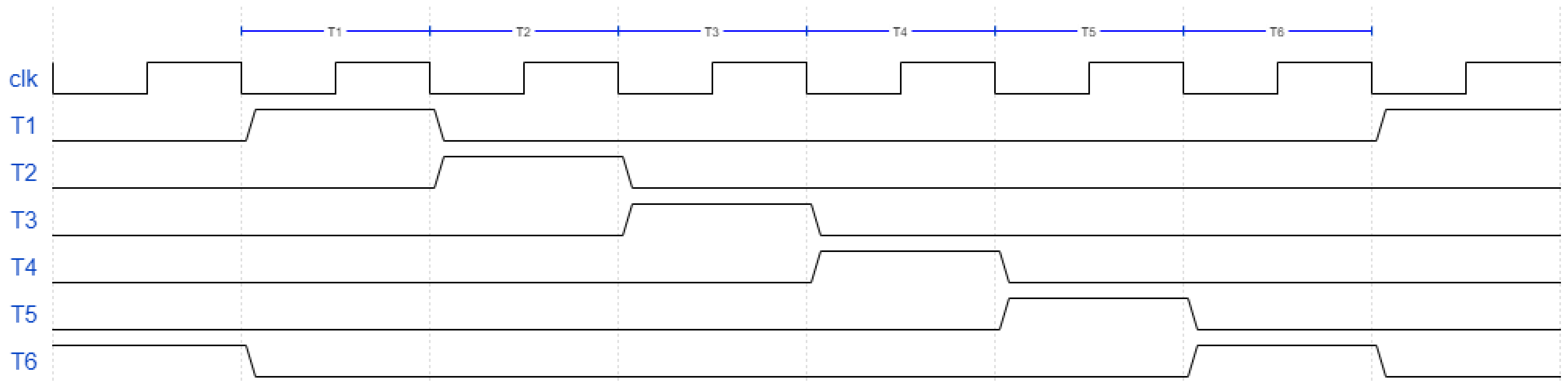
- The control unit generates the control words that fetch and execute each instruction
- The computer pass through different timing states during fetch and execute
- SAP-1 uses a 6-bit ring counter (circular shift register) to generate the timing states

$$T = T_6T_5T_4T_3T_2T_1$$

- The ring counter start from

$$T = 000001$$

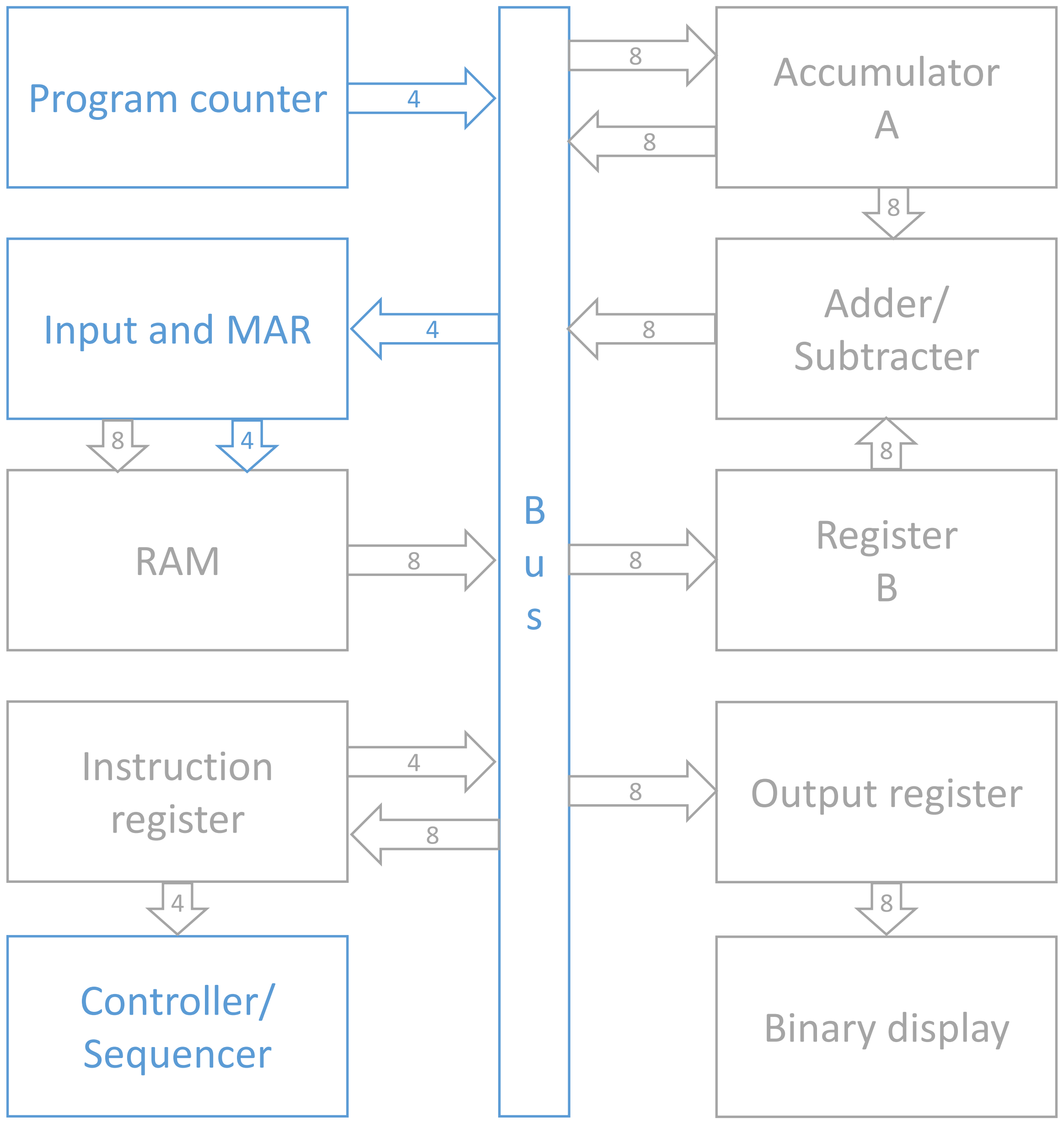
Timing states



- Each timing state starts and ends between two negative clock edges
- During the first timing state the first bit remains high, during the second the second bit remains high and so on
- The positive clock edge occurs in the middle of each timing states

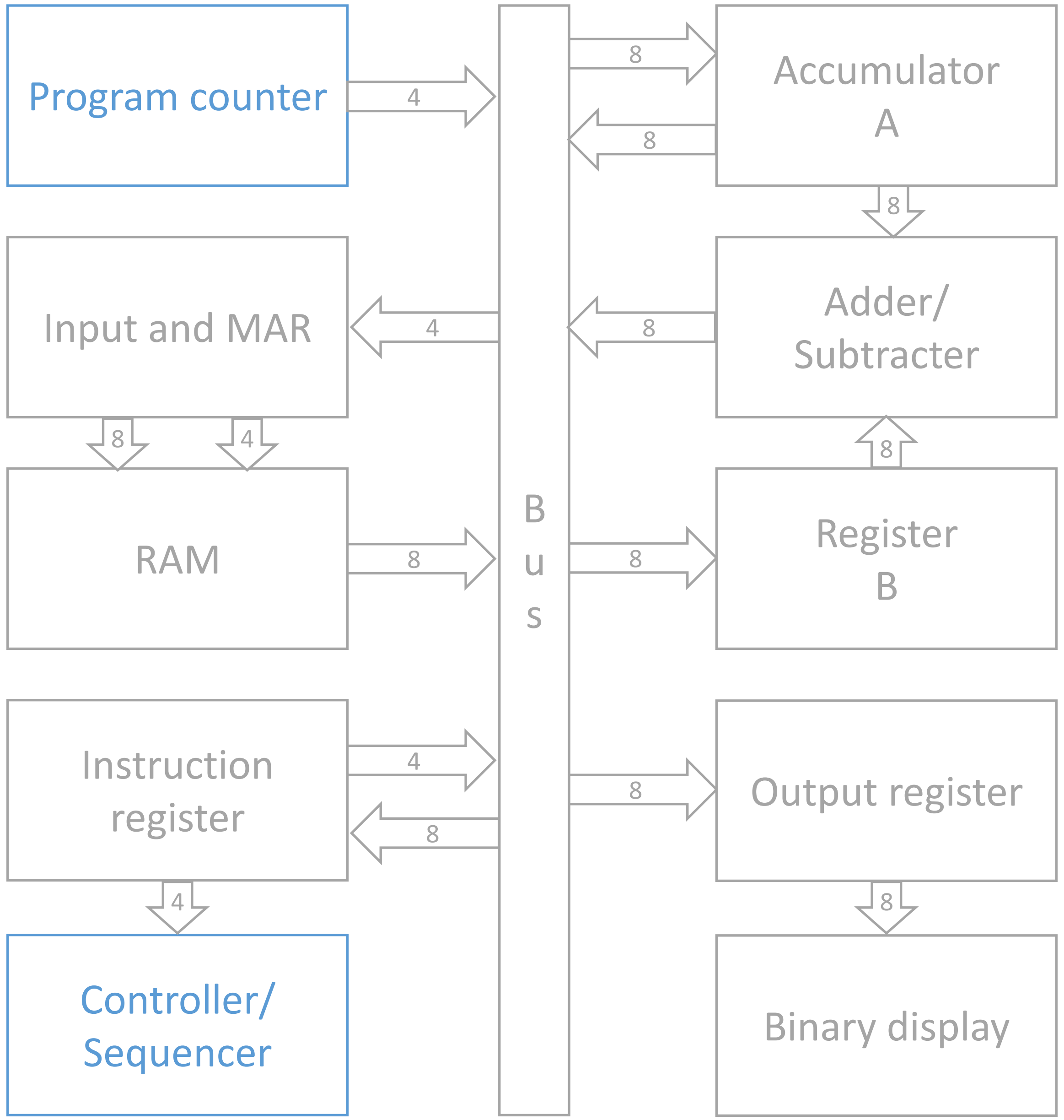
T_1: Address state

- The address in the program counter is transferred to the MAR



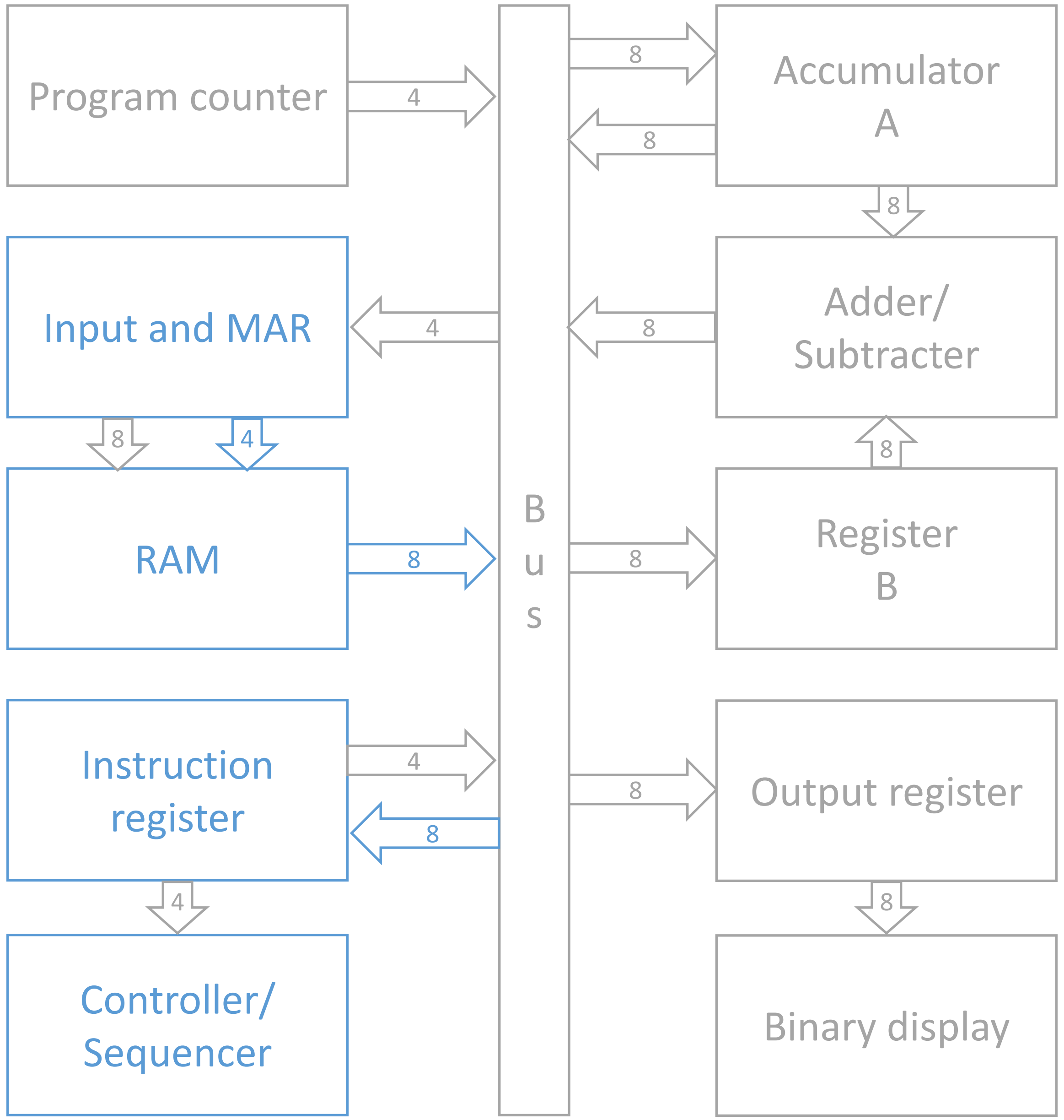
T_2: Increment state

- The program counter is incremented



T_3: Memory state

- The addressed RAM instruction is transferred from the memory to instruction register

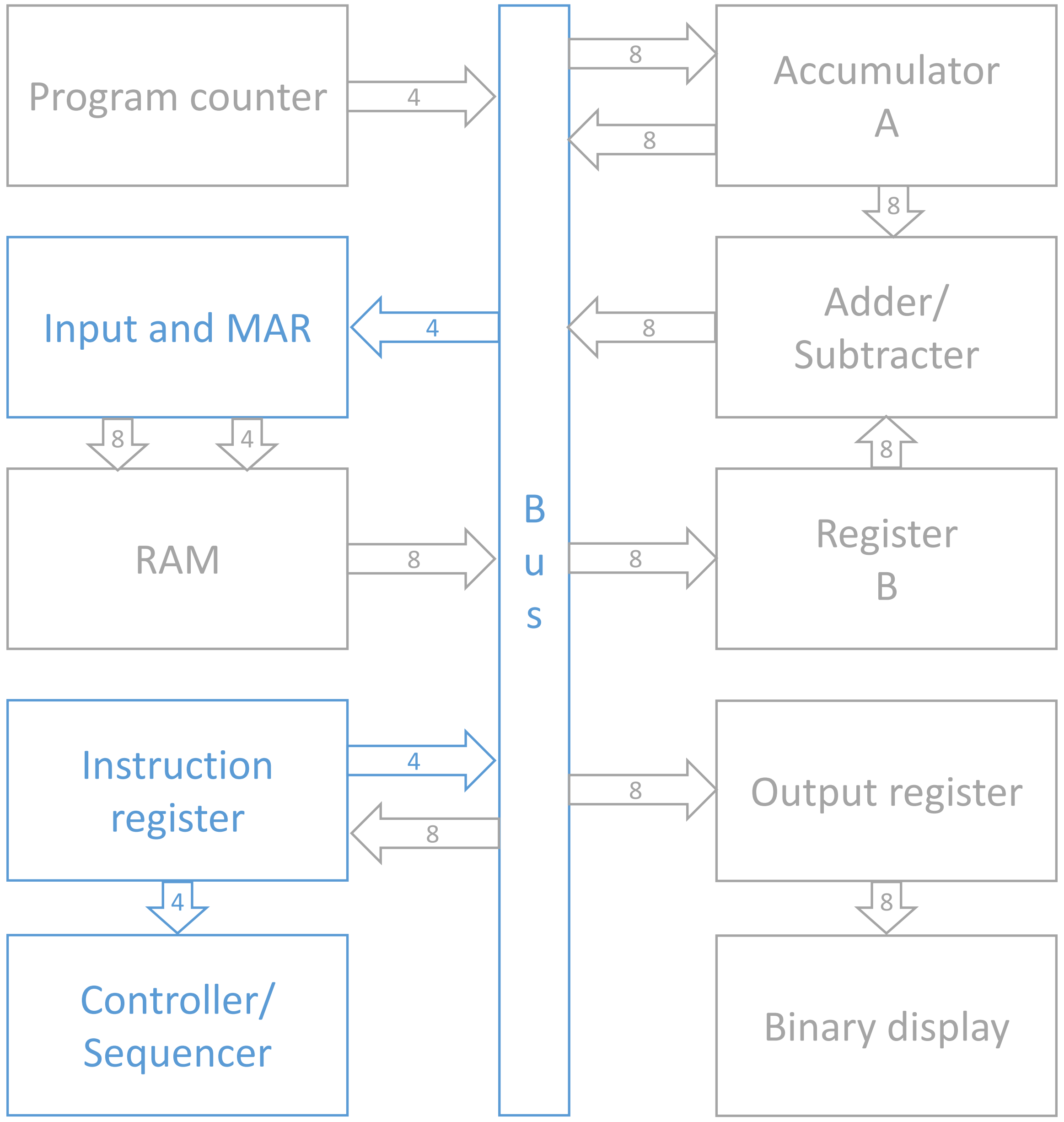


Timing states

- The first three timing states are called fetch cycle
- The second three timing states are called execution cycle
- The registers transfer during execution cycle are instruction dependent
- There is a different control routine for each instruction

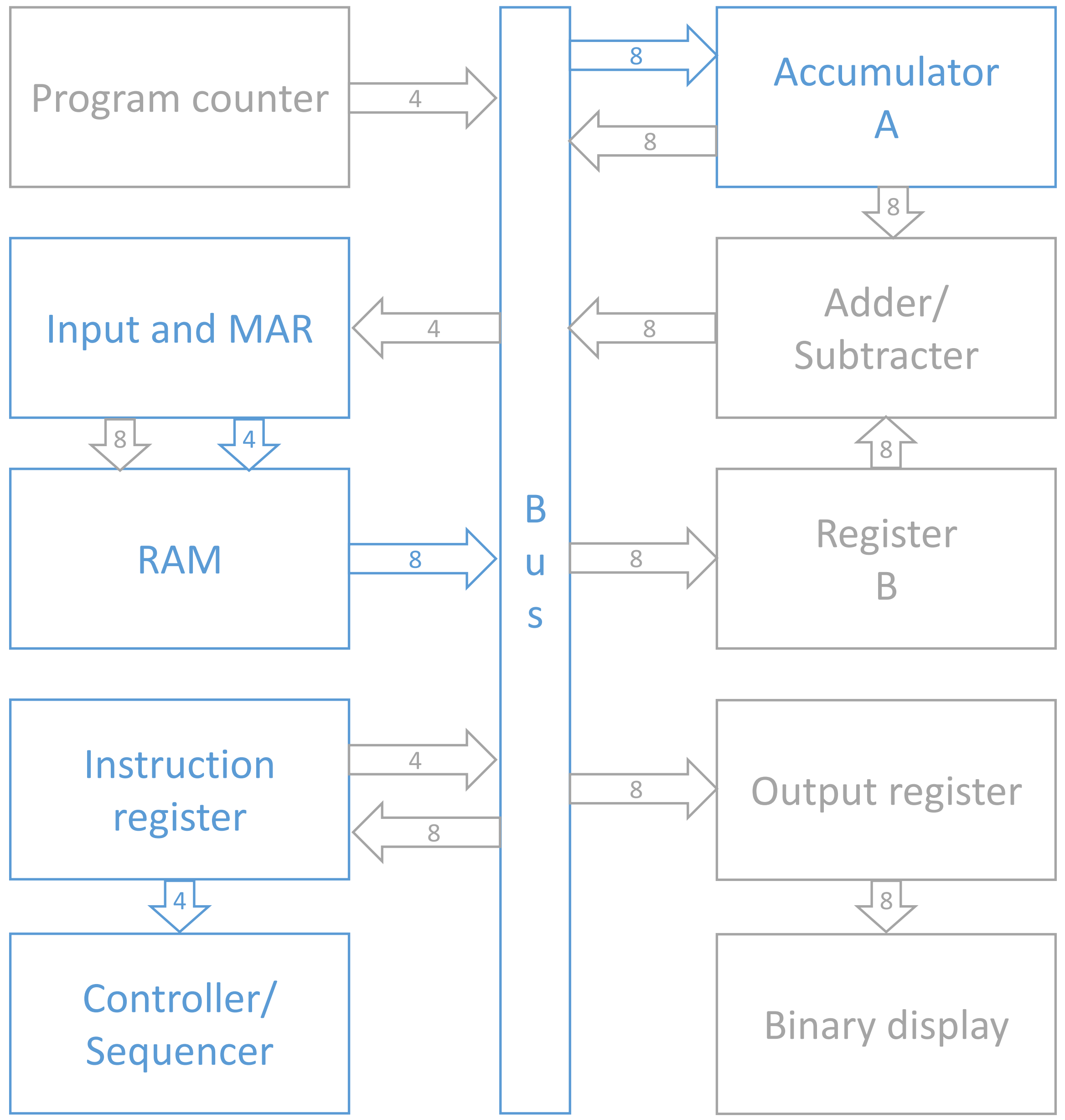
T_4 for LDA routine

- The first nibble of the instruction register (the instruction field) goes to the Controller/Sequencer
- The second nibble (the address field) is loaded into the MAR



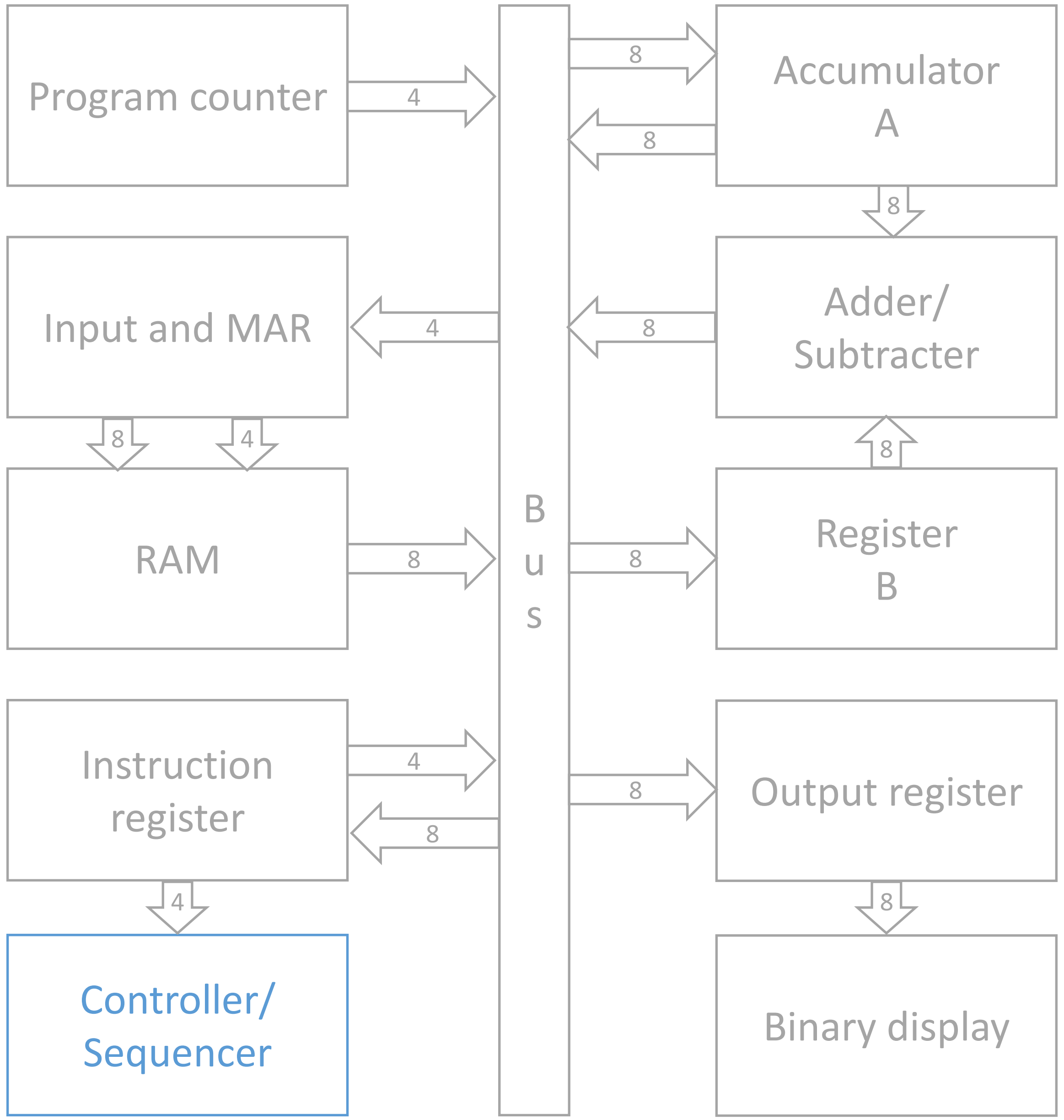
T_5 for LDA routine

- The addressed data word in RAM is copied to the accumulator



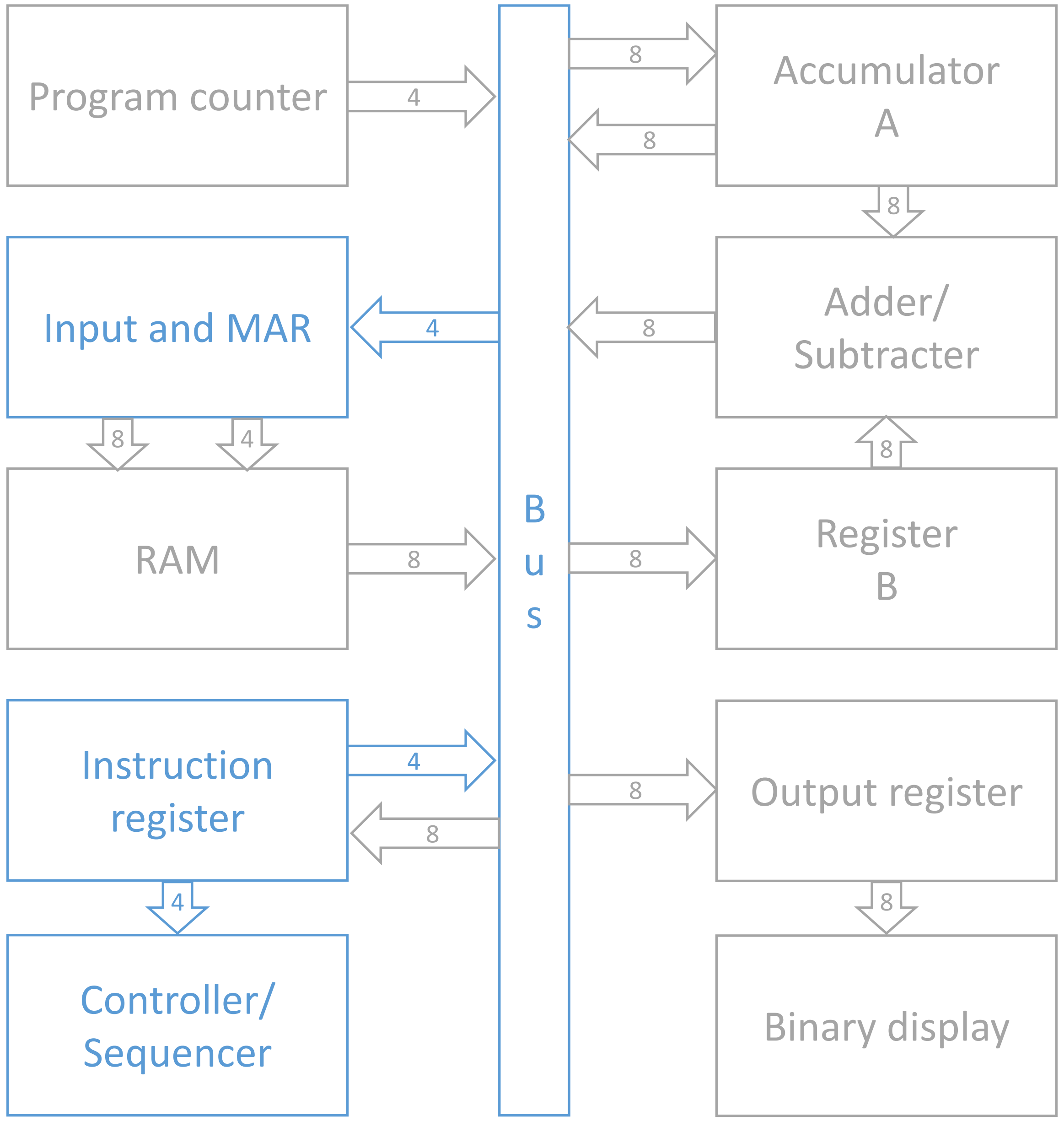
T_6 for LDA routine

- It is a nop (no-operation) state
- All registers are inactive



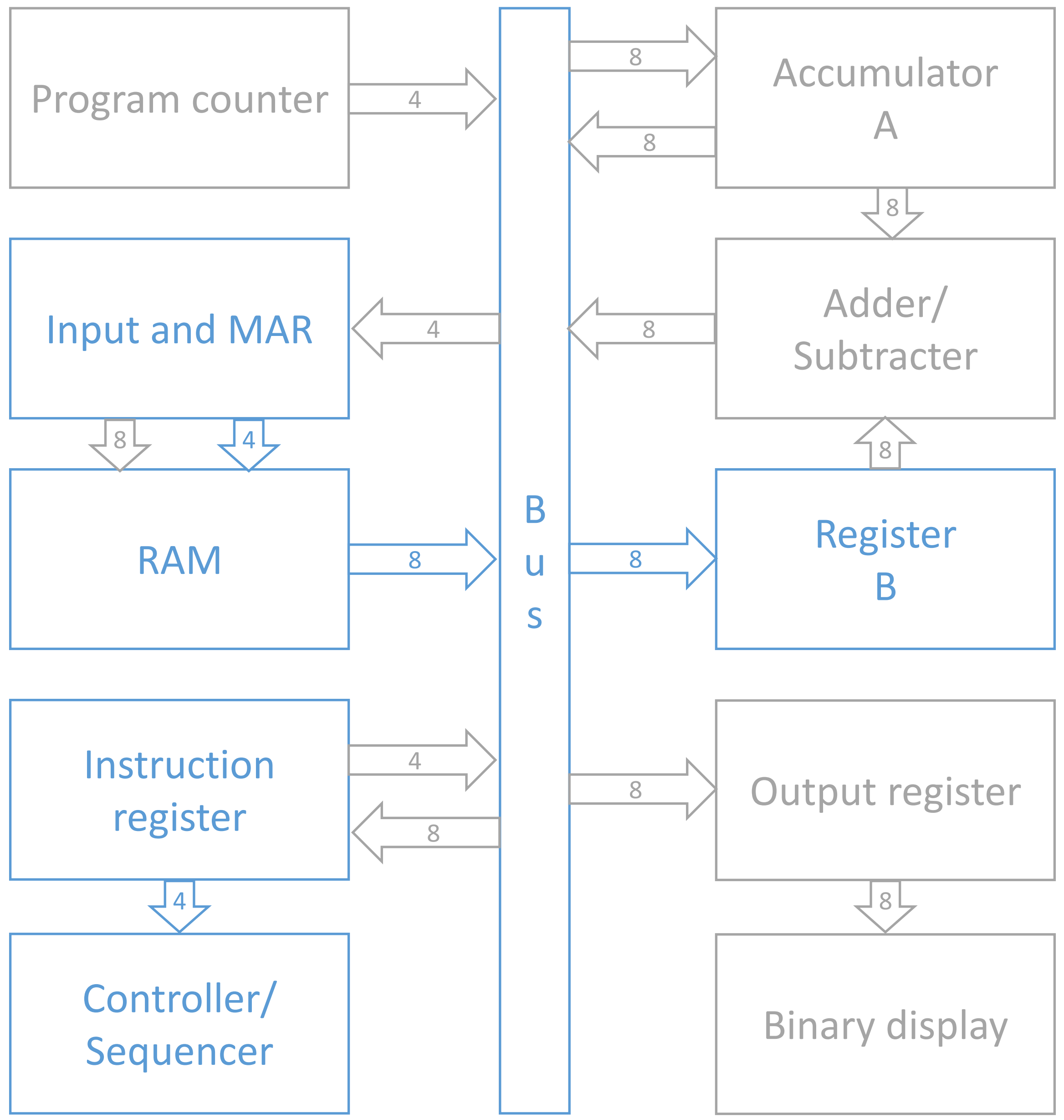
T_4 for ADD routine

- The first nibble of the instruction register (the instruction field) goes to the Controller/Sequencer
- The second nibble (the address field) is loaded into the MAR



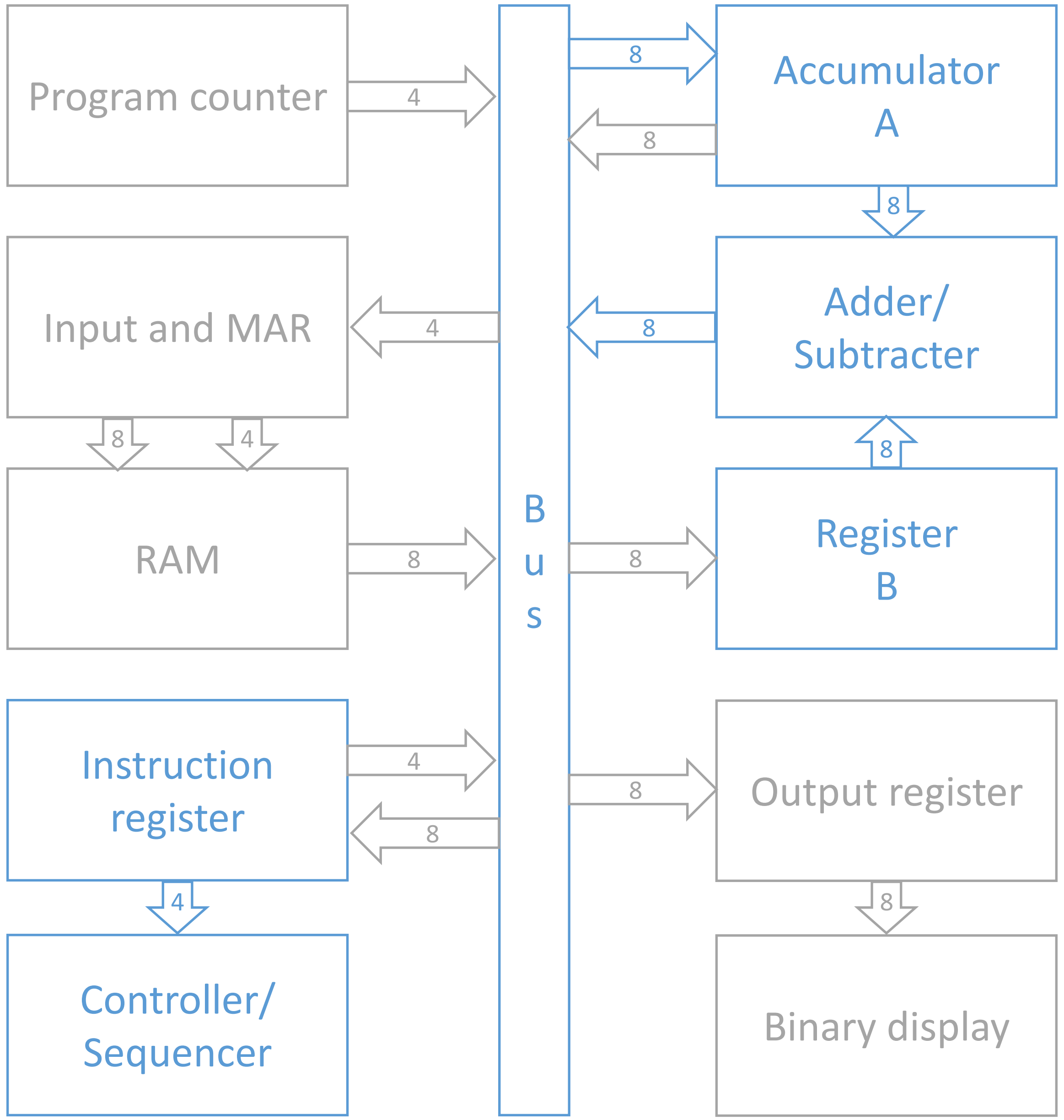
T_5 for ADD routine

- The addressed data word in RAM is copied to register B



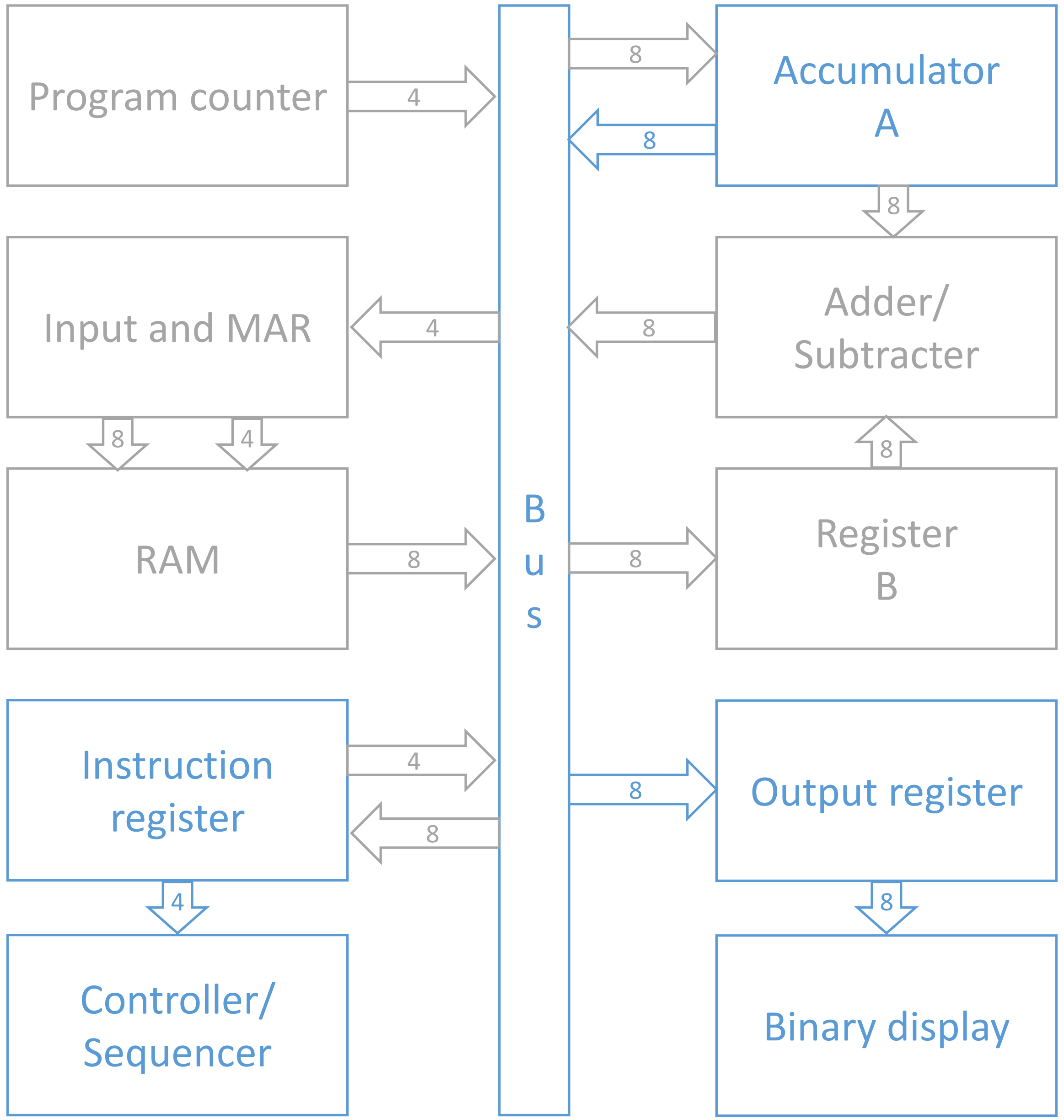
T_6 for ADD routine

- The output of the Adder/Subtracter is loaded to the accumulator
- Set up time and propagation delay prevent racing of the accumulator
- SUB control routine is identical except for the configuration of the Adder/Subtracter



T_4 for OUT routine

- The first nibble of the instruction register (the instruction field) goes to the Controller/Sequencer
- The accumulator content is copied to the output register
- The binary display shows its content
- T_5 and T_6 for the OUT control routine are nop

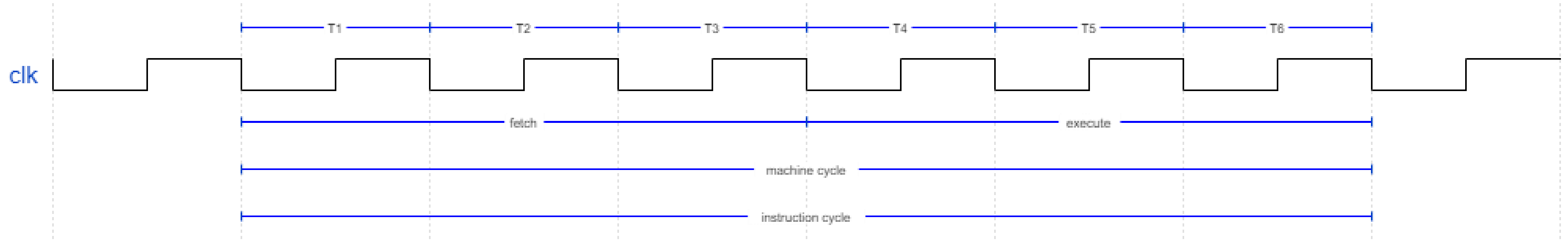


HLT routine

- HLT does not require a control routine because no registers are involved in its execution
- The Controller/Sequencer stops the computer by turning off the clock

Machine/Instruction cycle

- SAP-1 has six timing states (three fetch and three execute) that are called a machine cycle
- In SAP-1 all the instruction can be executed within a machine cycle -> the instruction cycle correspond to the machine cycle
- That is not always the case: the instruction cycle can use more than one machine cycle



Microprogram

- Controller/Sequencer send out one control word every state T, called microinstructions because it is part of an instruction
- With this nomenclature instructions are called macroinstructions
- In SAP-1 each macroinstruction is made by three microinstructions
- The fetch routine is also made by three microinstructions

Exercises

- Write an SAP-1 program that perform and display the result of the arithmetic operation 4+5-3
- If the clock frequency of SAP-1 is 4 MHz, how long does it take to run the previous program?