SAP Architecture:

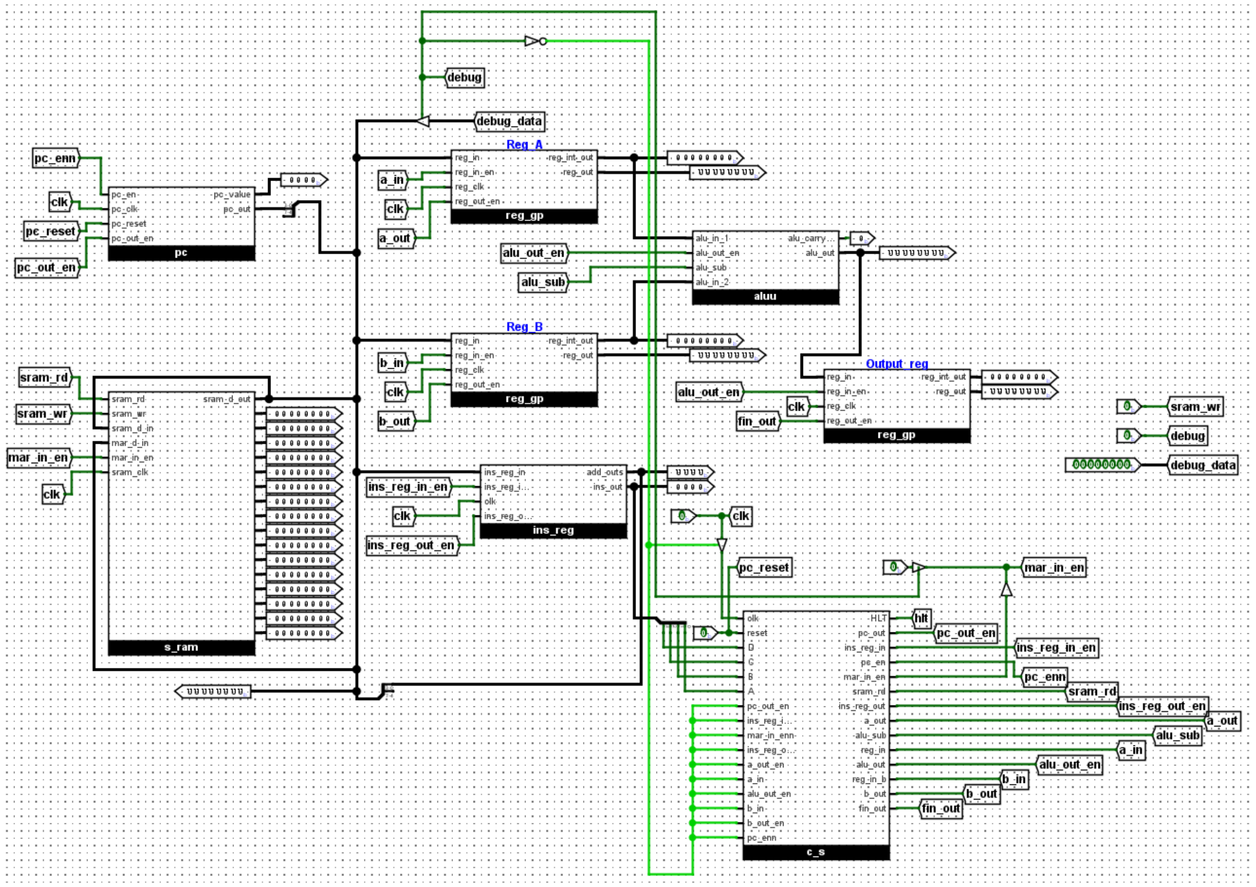


Figure 1: SAP architecture

The SAP-1 architecture consists of several components :

- **Program Counter (PC)**: Holds the address of the next instruction to be executed.

- **Memory (RAM)**: Holds the memory address of the data to be fetched from or written to memory. stores instructions and data.

- **Instruction Register (IR)**: Holds the current instruction to be executed.

- **Registers A and B**: General-purpose registers for holding temporary data.

- **Adder/Subtractor Unit(ALU)**: Performs addition and subtraction operations.

- **Output Register (OUT)**: Holds data to be sent to the output device.

- **Control Unit**: Generates the necessary control signals for executing instructions.

This was the designed control sequencer:

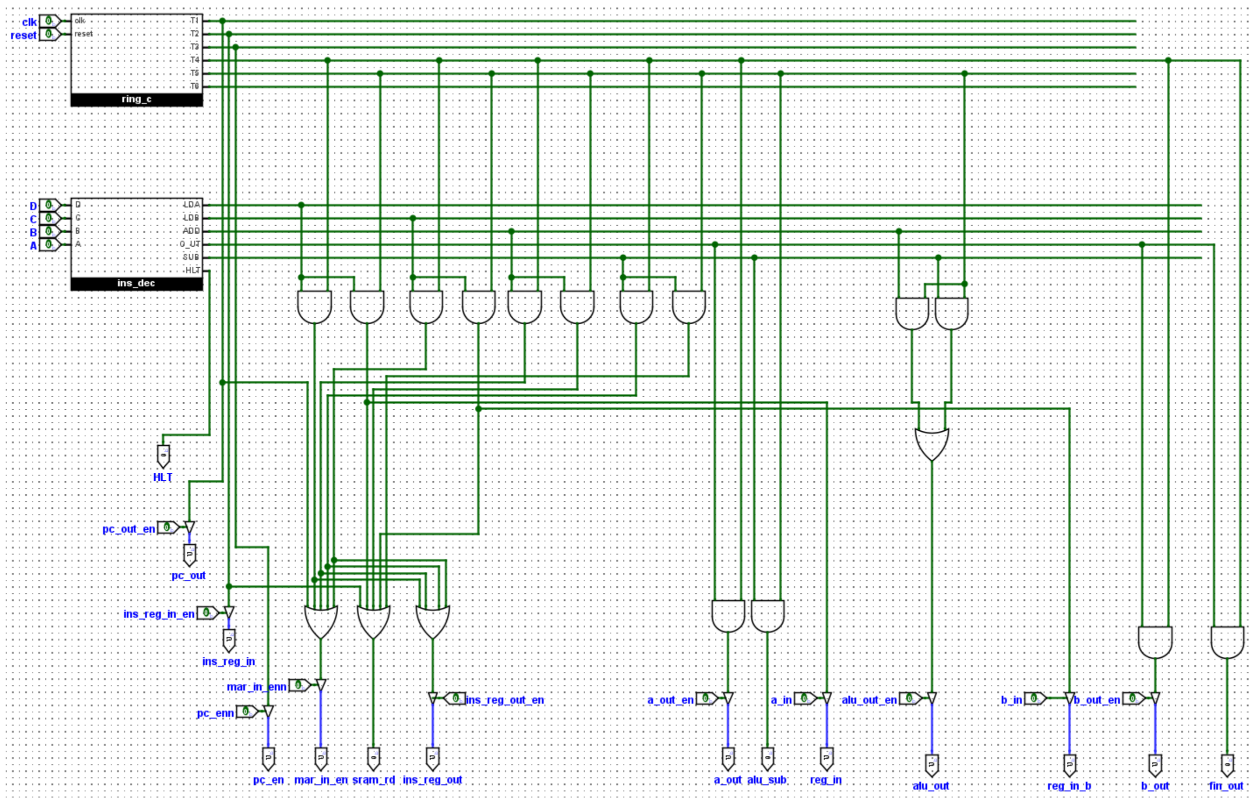


Figure 2: Control sequencer

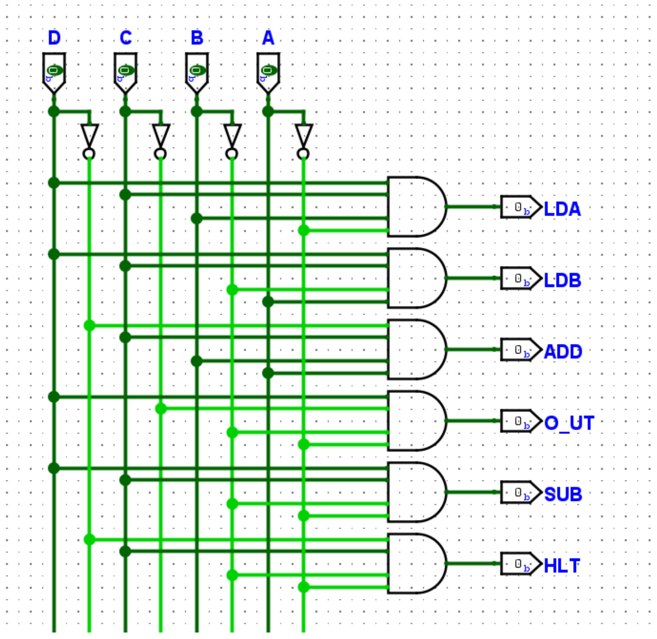


Figure 3: Instruction decoder

Instruction Set:

The instruction set defines the operations that SAP-1 can perform. This design can perform the following instructions : LDA, LDB, ADD, SUB, HLT, and OUT.

Instruction Format:

Each instruction is 8 bits long, divided into two parts:

- **Opcode** (4 bits): Specifies the operation to be performed.
- **Operand** (4 bits): Specifies the address or data for the operation.

Instructions:

1. **LDA (Load A Register)**
    - **Opcode**: 0001
    - **Description**: Load the contents of the memory address specified by the operand into the A register.
2. **LDB (Load B Register)**
    - **Opcode**: 0010
    - **Description**: Load the contents of the memory address specified by the operand into the B register.
3. **ADD (Add)**
    - **Opcode**: 0111
    - **Description**: Add the contents of the A and B registers
4. **SUB (Subtract)**
    - **Opcode**: 1100
    - **Description**: Subtract the contents of the B register from the A register
5. **HLT (Halt)**
    - **Opcode**: 0100
    - **Description**: Halt the execution of the program.
6. **OUT (Output)**
    - **Opcode**: 1000
    - **Description**: Output the contents of ALU to the output register.

Control signals are generated by the control sequencer to coordinate the execution of instructions. Here's a step-by-step breakdown of the control sequencer operation for each instruction:

Load register A(LDA):

LDA 01(0001 0001)

- At first some data is saved in the memory location 1. For this, the steps are followed: Debug on > pc_reset is pulsed> Debug data = 00000000 > mar_in_en on> clk pulsed > mar_in_en off > Debug data=00010001> sram_wr on > clk pulsed > sram_wr off > debug data= 00000001 >mar_in_en on> clk pulsed > mar_in_en off > debug data= 00000100 > sram_wr on > clk pulsed > sram_wr off > debug off.

Now the data 00000100 is saved to memory location 1.

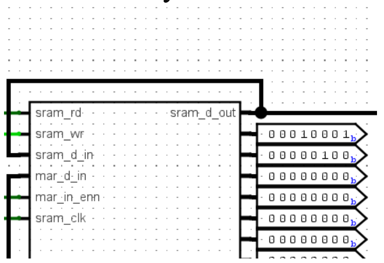


Figure 4: Data saved in memory location 1

To perform LDA 01 operation clk is pulsed. For each pulse, T state is incremented by 1. There are in total 6 T-states.

Fetch (T1 - T3):

T1: ‘mar_in_en’ and ‘pc_out_en’ are enabled.
T2: ‘sram_rd’ and ‘ins_reg_in_en’ are enabled.
The memory location is saved in the instruction register(0001)
T3: ‘pc_out_en’ is enabled.
Program counter was incremented by 1.

Execute(T4 - T6):

T4: ‘mar_in_en’ and ‘ins_reg_out_en’ are enabled.
T5: ‘sram_rd’ and ‘a_in’ are enabled.
The data of the address saved in the instruction register is saved in the register A.
T6: Nothing is executed here.

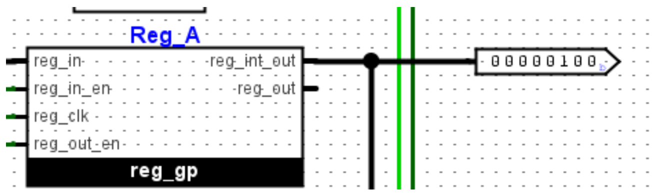
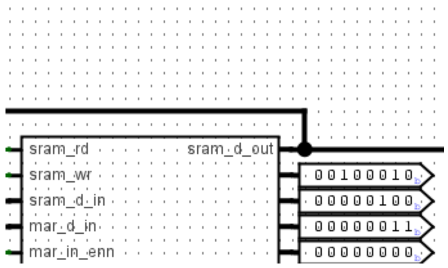


Figure 4: Data saved in register A

Load register B(LDB):

LDB 02(0010 0010)

- At first, some data is saved in the memory location 2. For this, the steps are followed: Debug on > pc_reset is pulsed >Debug data = 00000000 > mar_in_en on> clk pulsed > mar_in_en off > Debug data=00100010> sram_wr on > clk pulsed > sram_wr off > debug data= 00000010 > mar_in_en on> clk pulsed > mar_in_en off > debug data= 00000011 > sram_wr on > clk pulsed > sram_wr off> debug off.
  Now the data 00000011 is saved to memory location 2.
  Here 0010 was used for LDB opcode and 0010 was the memory location.



To perform LDB 02 operation, clk is pulsed. For each clk pulse, T state is incremented by 1. There are in total 6 T-states. T1 – T6 are as same as the LDA instruction.

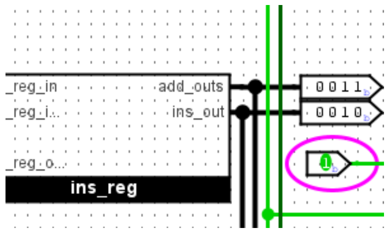


Figure 5: Opcode and operand in Instruction register

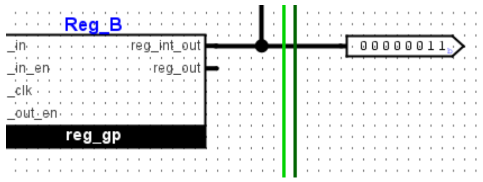


Figure 6 : Data saved in register B

ADD:

ADD A,B(01110000)

- To add the data stored in register A and B, the instruction is stored in memory location 0. For this, the steps are followed:
  Debug on > pc_reset is pulsed >Debug data = 00000000 > mar_in_en on> clk pulsed > mar_in_en off > Debug data=01110000> sram_wr on > clk pulsed > sram_wr off > debug off.
  Now the instruction 01110000 is saved to memory location 0.
  Here 0111 was used for ADD opcode .

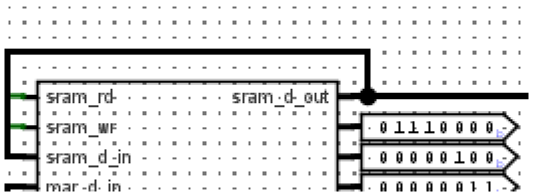
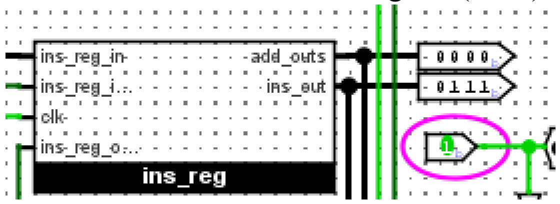


Figure 7: Instruction for ADD at memory location 0

Now 'clk' is pulsed for performing addition.

Fetch (T1 - T3):

T1: ‘mar_in_en’ and ‘pc_out_en’ are enabled.
T2: ‘sram_rd’ and ‘ins_reg_in_en’ are enabled.
The memory location is saved in the instruction register(0111)



T3: ‘pc_out_en’ is enabled.
Program counter was incremented by 1.

Execute(T4 - T6):

T4: ‘mar_in_enn’ and ‘ins_reg_out_en’ are enabled.
T5: ‘sram_rd’, ‘alu_out_en’ are enabled.
The data of the address saved in the instruction register is saved in the register A. The addition of 2 data is performed seen in the ALU output.
T6: This state is unused for this instruction.

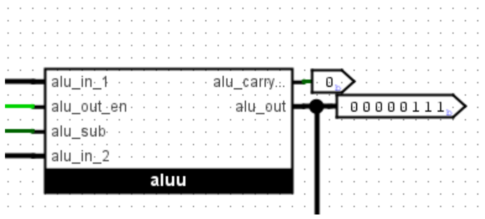


Figure 8: ADD operation in ALU

SUB:

SUB A,B(11000000)

- To substract the data stored in register B from A, the instruction is stored in memory location 0. For this, the steps are followed:
  Debug on > pc_reset is pulsed >Debug data = 00000000 > mar_in_en on> clk pulsed > mar_in_en off > Debug data=11000000> sram_wr on > clk pulsed > sram_wr off > debug off.
  Now the instruction 11000000 is saved to memory location 0.
  Here 1100 was used for SUB opcode .

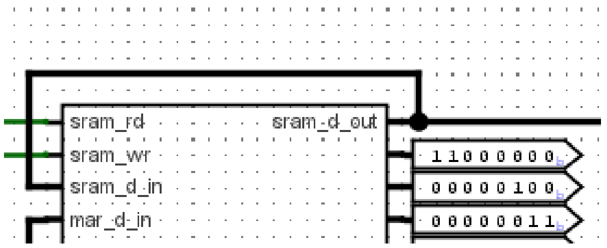
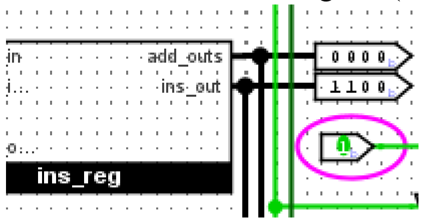


Figure 9: Instruction for SUB at memory location 0

Fetch (T1 - T3):

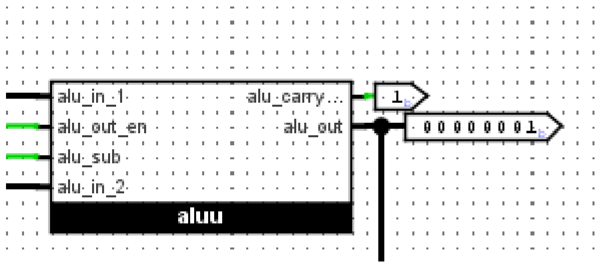
T1: 'mar_in_en' and 'pc_out_en' are enabled.
T2: 'sram_rd' and 'ins_reg_in_en' are enabled.
The memory location is saved in the instruction register(1100)



T3: 'pc_out_en' is enabled.
Program counter was incremented by 1.

Execute(T4 - T6):

T4: 'mar_in_en' and 'ins_reg_out_en' are enabled.
T5: 'sram_rd', 'alu_out_en' , 'alu_sub' are enabled.
The subtraction between 2 data is performed seen in the ALU output.
T6: This state is unused for this instruction.



Here alu_carry =1 represented that , the subtraction resulted a positive number.

OUT:

OUT(1000000)

To show the data from the alu output to the output register, the instruction for output is stored in memory location 0. For this, the steps are followed:

> Debug on > pc_reset is pulsed >Debug data = 00000000 > mar_in_en on> clk pulsed > mar_in_en off > Debug data=1000000> sram_wr on > clk pulsed > sram_wr off > debug off .
> Now the instruction 10000000 is saved to memory location 0.
> Here 1000 was used for OUT opcode .

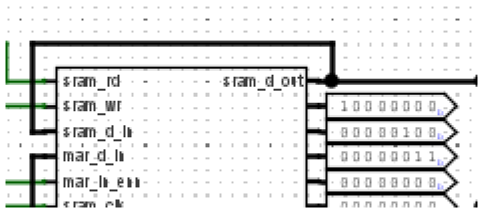


Figure 10: Instruction for OUT at memory location 0

Fetch (T1 - T3):

T1: 'mar_in_en' and 'pc_out_en' are enabled.
T2: 'sram_rd' and 'ins_reg_in_en' are enabled.
The memory location is saved in the instruction register(1000)
T3: 'pc_out_en' is enabled.
Program counter was incremented by 1.

Execute(T4 - T6):

T4: 'a_out' , 'b_out' and 'fin_out' are enabled.
The output of ALU is saved in output register. Value stored in Register A, Register B and the result of the arithmetic operation are displayed.
T5: This state is unused for this instruction.
T6: This state is unused for this instruction.

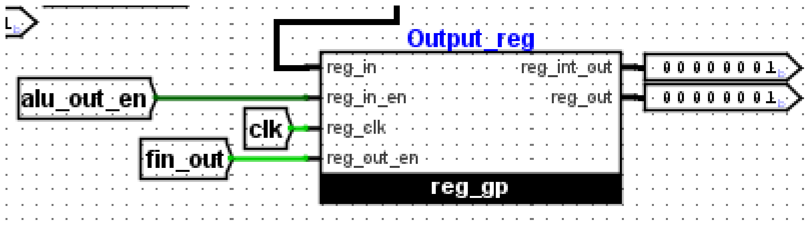
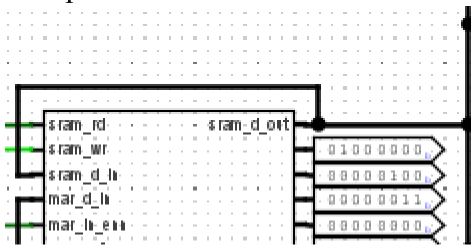


Figure 11: Final output of substruction operation

HLT:

HLT(01000000)

- To halt the current exicution, the instruction is stored in memory location 0. For this, the steps are followed:
- Debug on > pc_reset is pulsed >Debug data = 00000000 > mar_in_en on> clk pulsed > mar_in_en off > Debug data=0100000> sram_wr on > clk pulsed > sram_wr off > debug off .
  Now the instruction 01000000 is saved to memory location 0.
  Here 0100 was used for HLT opcode .



Fetch (T1 - T3):

T1: 'mar_in_enn' and 'pc_out_en' are enabled.
T2: 'sram_rd' and 'ins_reg_in_en' are enabled.
The memory location is saved in the instruction register(0100)
T3: 'pc_out_en' and 'hlt' are enabled.
Program counter was incremented by 1.

Execute(T4 - T6):

T4: 'hlt' is enabled.
T5: 'hlt' is enabled.
T6: 'hlt' is enabled

Mainly after the HLT instruction execution no other execution happened or no other pins enabled. Thus it halted the other operation.

These are the instructions for the designed SAP architecture. Following these instructions, the desired operation can be operated through this architecture.