- DEUNIAC REPORT -

Control Function & Microoperations Table

FETCH >>> T0: IR ← IM[PC]	
DECODE >>> T1: PC←PC+1, Q←IR (10), D0, D1...D15 ←Decode IR (9-6)	
DBL	D_0T_3:R_d ← shl R_{S1}, SC←0
DBT	D_1T_3:R_d ← shr R_{S1}, SC←0
ADD	D_2T_3:R_d ← R_{S1} + R_{S2}, SC←0
INC	D_3T_3:R_d ← R_{S1} +1, SC←0
AND	D_4T_3:R_d ← R_{S1} ∧ R_{S2}, SC←0
NOT	D_5T_3: R_d ← R_{S1}', SC ← 0
XOR	D_6T_3:R_d ←R_{S1} ⊕ R_{S2}, SC←0
HLT	D_7T_3: SC_en ← 0
ST	D_8T_3Q': AR ← IR(3-0) D_8T_4Q': DM[AR] ← Rd, SC ← 0 D_8T_3Q: AR ← Rd D_8T_4Q: DM[AR] ← S_1S_2, SC ← 0
LD	D_9T_3Q': R_d ← S_1S_2, SC ← 0 D_9T_3Q: AR ← IR(3-0) D_9T_4Q: DM_read ← 1 D_9T_5Q: D_9T_6Q: R_d ← DM[AR], SC ← 0
IO	$D_{10}T_3$Q': OutR ← R_{S1}, SC ← 0 $D_{10}T_3$Q: R_d ← InpR, SC ← 0
TSF	$D_{11}T_3$: R_d ← R_{S1}, SC ← 0
JMP	$D_{12}T_3$Q': PC ← IR (4-0), SC ← 0 $D_{12}T_3$QV: PC ← IR (4-0), SC ← 0
CAL	$D_{13}T_3$: SM[SP] ← PC $D_{13}T_4$: SP ← SP + 1, PC ← IR (4-0), SC ← 0
RET	$D_{14}T_3$: SP ← SP - 1 $D_{14}T_4$: SM_read ← 1 $D_{14}T_5$: $D_{14}T_6$: PC ← SM [SP], SC ← 0
JMR	$D_{15}T_3$: PC ← PC + S1S2, SC ← 0

Write Selection Table (4 Bits)

Sel_write[2..0]	SELECTION
000	ALU_out
001	Rd (R0-R1-R2)
010	S1S2
011	DM[AR]
100	S1 (R0-R1-R2)
101	INPR

I used the output "sel_write[2..0]" for selection of writeable data in the bus system. A MUX can control the data. This data can be load into *AR*, *Rd*, *DM[AR]*, *OUTR*.

Program Counter Selection Table (5 Bits)

Sel_PC[1..0]	SELECTION
00	SM[SP]
01	PC
10	Address
11	PC+offset

I used the output "sel_PC[1..0]" for selection of writeable data for program control in the bus system. A MUX can control the data. This data can be load into *PC*, *SM[SP]*.

Load Selection Table (in BUS)

Signal Name	Load into
LD_AR	AR
LD_PC	PC
LD_Rd	Rd
LD_IR	IR
DMW	Data Memory[AR]
SMW	Stack Memory[SP]
In	INPR
Out	OUTR

Using these signals I can insert writable data into fields.

Assembly Code

```
          ORG I 0         ;Origin of instruction memory
0         LD R0, @A       ;Load operand from address A to R0
I1        LD R1, @C       ;Load counter from location C to R1
I2        INC R2, R1      ;Increment R1 (counter) and store to R2
I3        JMP 8, Q       ;If v= 1 jump to HLT and finish the program
I4        TSF R1, R2      ;Transfer counter value to R1.
I5        ADD R2, R0, R1 ;Add R0 and R1, store in R2
I6        TSF R0, R2      ;Transfer R2 (sum) to R0
I7        JMR -5          ;Jump to beginning of loop
I8        HLT             ;Halt computer
          ORG D 0         ;Origin of data segment
D0   A:   DEC 2           ;Decimal value 2
D1   C:   DEC 12          ;Decimal value 12
```

Assembly Code

INSTRUCTION NUMBER	INSTRUCTION NAME	BINARY
I0	LD	1 1001 00 0000
I1	LD	1 1001 01 0001
I2	INC	X 0011 10 01 XX
I3	JMP (to I8)	1 1100 X 01000
I4	TSF	X 1011 01 10 XX
I5	ADD	X 0010 10 00 11
I6	TSF	X 1011 00 10 XX
I7	JMR (-6)	X 1111 XX 1110
I8	HLT	X 0111 XX XX XX

Assembly to binary

Since I initially increased the PC during Decode-Fetch in the Control unit, I took the situation under control by making -6 instead of -5 in the JMR process.

Loops

Loop 0

NUMBER	INSTRUCTION	CHANGE
I0	LD	R0
I1	LD	R1
I2	INC	R2
I3	JMP (to I8)	X
I4	TSF	R1
I5	ADD	R2
I6	TSF	R0
I7	JMR (-6)	X

Loop 1

NUMBER	INSTRUCTION	CHANGE
I2	INC	R2
I3	JMP (to I8)	X
I4	TSF	R1
I5	ADD	R2
I6	TSF	R0
I7	JMR (-6)	X

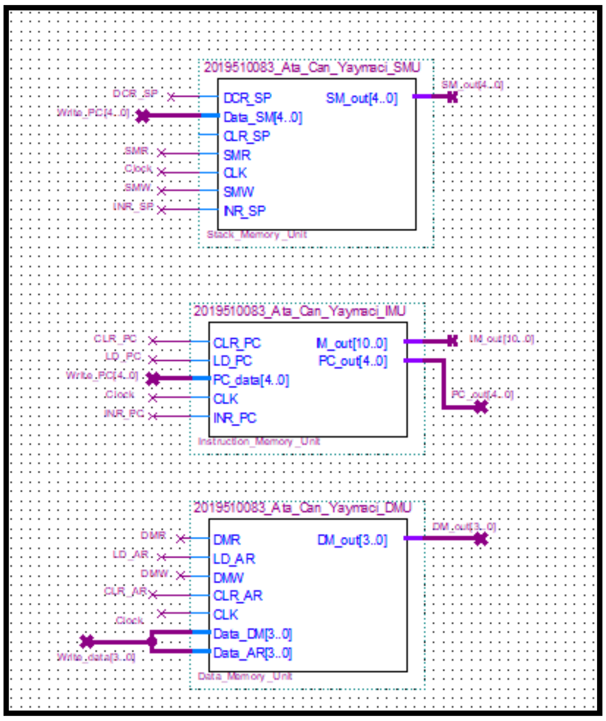
Loop 2

NUMBER	INSTRUCTION	CHANGE
I2	INC	R2
I3	JMP (to I8)	X
I4	TSF	R1
I5	ADD	R2
I6	TSF	R0
I7	JMR (-6)	X

Loop 3

NUMBER	INSTRUCTION	CHANGE
I2	INC	R2
I3	JMP (to I8)	X
I8	HLT	Halt comp.

Design and Improvements

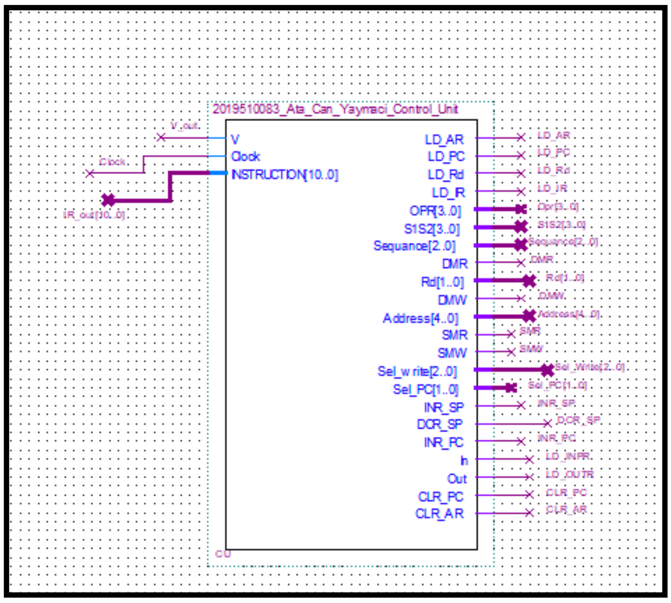


Memory Units

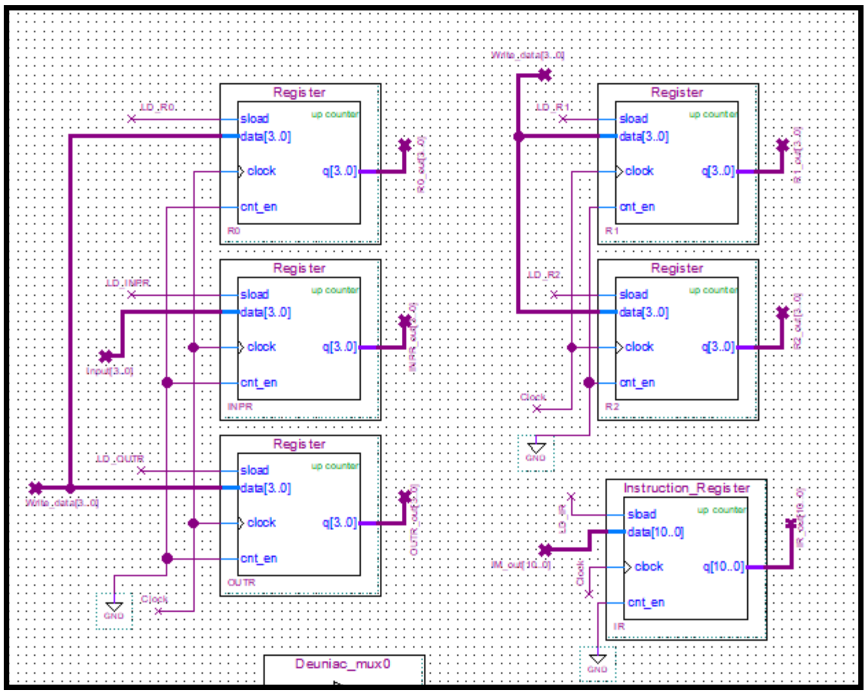
SMU is “Stack Memory unit” with a Stack Pointer and a Stack Memory(5x16).

IMU is “Instruction Memory unit” with a Program Counter and an Instruction Memory(11x32).

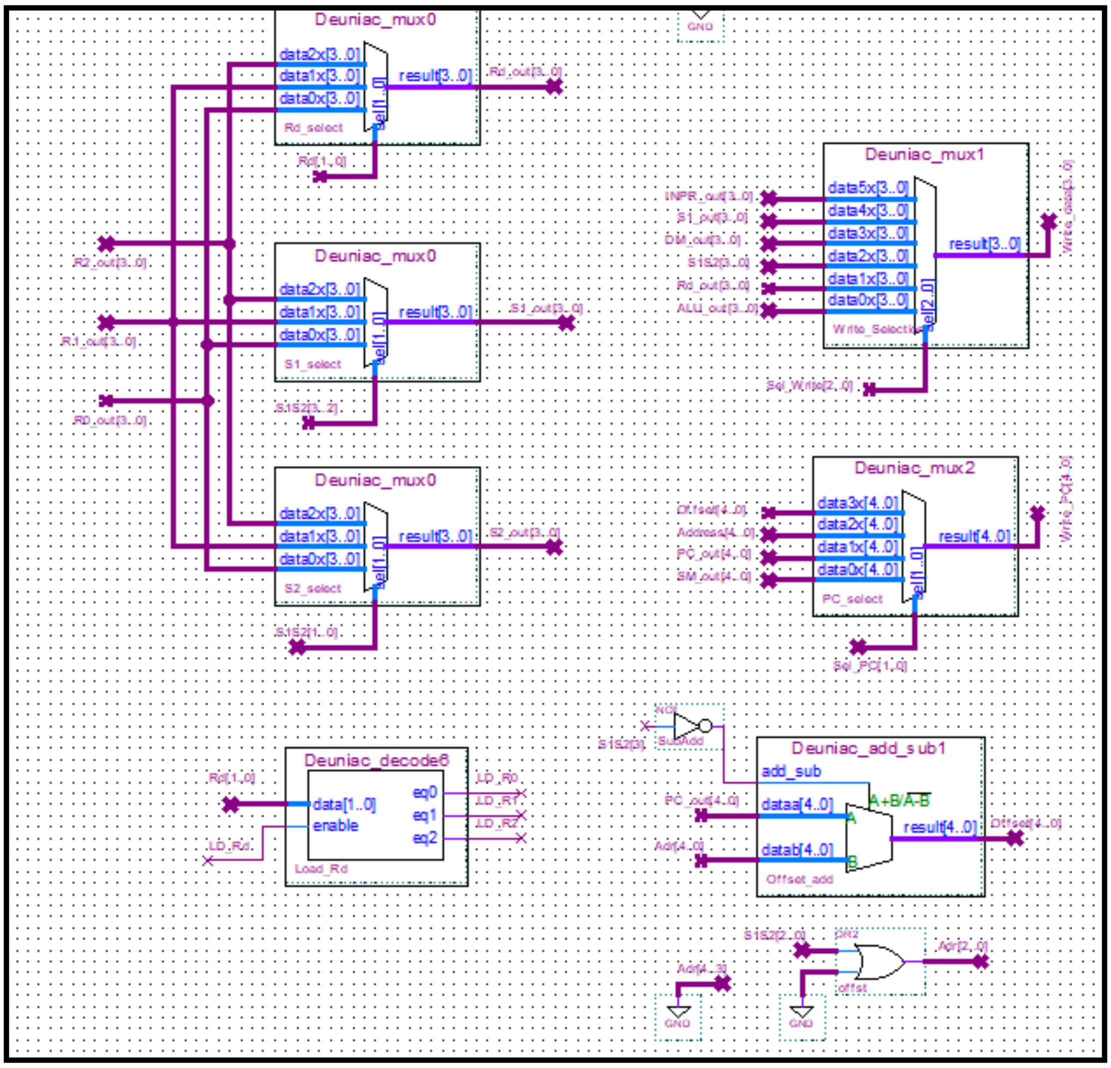
DMU is “Data Memory unit” with an Address Register and a Data Memory(4x16).



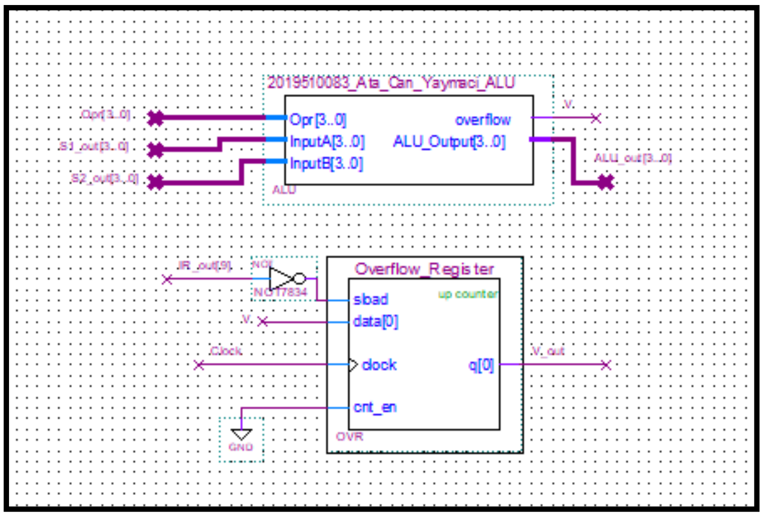
Control unit



Registers (R0/R1/R2/INPR/OUTR/IR)



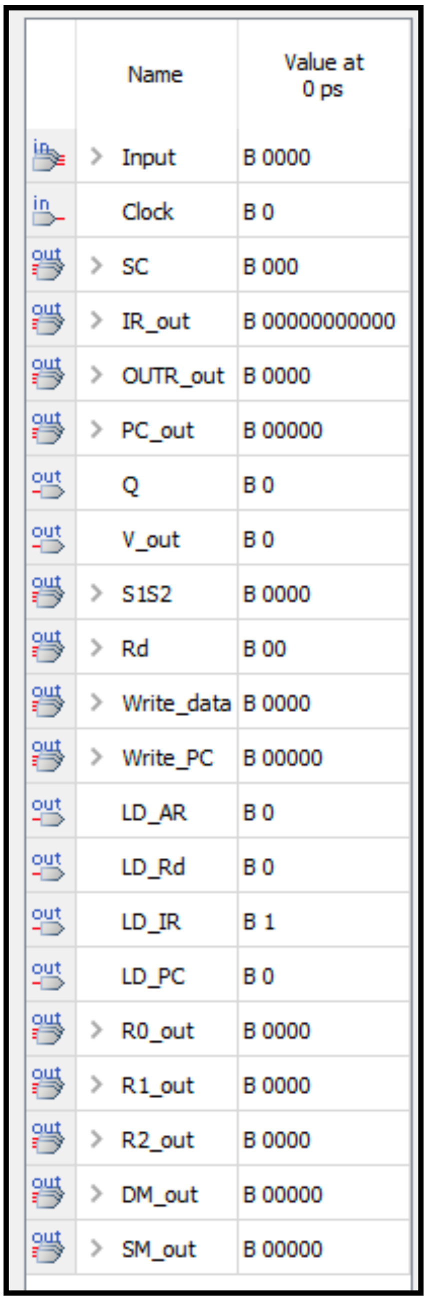
Selections and an offset calculator



ALU and overflow register

Since overflows are not stored correctly during checks (JMP), I saved the overflow with a register.

Waveform



Name	Value at 0 ps
Input	B 0000
Clock	B 0
SC	B 000
IR_out	B 00000000000
OUTR_out	B 0000
PC_out	B 00000
Q	B 0
V_out	B 0
S1S2	B 0000
Rd	B 00
Write_data	B 0000
Write_PC	B 00000
LD_AR	B 0
LD_Rd	B 0
LD_IR	B 1
LD_PC	B 0
R0_out	B 0000
R1_out	B 0000
R2_out	B 0000
DM_out	B 00000
SM_out	B 00000

Waveform headers

Input: Input for INPR

Clock: Clock for system

SC: System counter (T0/T1/T2/T3/T4/T5/T6)

IR_out: Instruction register

OUTR_out: Output register

PC_out: Program counter

Q: Most significant bit in instruction

S1S2: IR(3..0)

Rd: IR(5..4) selects R0/R1/R2

Write_data: Selection for writeable data (with a 4bit mux)

Write_PC: Selection for writeable data (with a 5bit mux)

LD_AR: Load signal for AR

LD_IR: Load signal for IR

LD_PC: Load signal for PC

R0_out: R0 register

R1_out: R1 register

R2_out: R2 register

DM_out: DM[AR] (Data memory)

SM_out: SM[SP] (Stack memory)