INFO-0012: Computation Structures

β-machine - Report

Maxime Goffart
180521

Olivier Joris
182113

Academic year 2020 - 2021

Control logic

We had to implement the instructions `ADDC`, `AND`, `CMPLEC`, `LD`, and `BNE` inside the control logic[1]. Also, we implemented the instruction `ST` which allows us to test the instruction `LD` more easily.

Instruction	Address	ALUFN	WERF	BSEL	WDSEL	WR	RA2SEL	PCSEL	Hexadecimal
ADDC	110000	0001	1	0	01	0	X(0)	00	190
AND	101000	0101	1	1	01	0	0	00	5D0
CMPLEC	110110	1110	1	0	01	0	X(0)	00	E90
LD	011000	0001	1	0	10	0	X(0)	00	1A0
BNE	011110	X(0000)	1	X(0)	00	0	X(0)	¬z	080
ST	011001	0001	0	0	X(00)	1	1	00	10C

Instruction memory

The instructions we implemented in order to test the instructions `ADDC`, `AND`, `CMPLEC`, `LD`, and `BNE`[2]:

Instruction	Hexadecimal	Effect	Instruction	Hexadecimal	Effect
ADDC(R31, 5, R0)	C01F0005	R0=5	CMPLEC(R1, -5, R29)	DBA1FFFB	R29=1
ADDC(R31, -5, R1)	C03FFFFB	R1= −5	CMPLEC(R1, -4, R28)	DB81FFFC	R28=1
ADDC(R0, 5, R2)	C0400005	R2=10	ADDC(R31, 0, R0)	C01F0000	R0=0
ADDC(R1, 5, R3)	C0610005	R3=0	ADDC(R31, 21, R30)	C3DF0015	R30=21
ADDC(R1, -5, R28)	C381FFFB	R28= −10	ST(R30, 0, R0)	67C00000	MEM[Reg[R0]] ←Reg[R30]
ADDC(R31, 12, R0)	C01F000C	R0=12	LD(R0, 0, R1)	60200000	Reg[R1] ←Mem[Reg[R0]]
ADDC(R31, 10, R1)	C03F000A	R1=10	ADDC(R31, 4, R0)	C01F0004	R0=4
AND(R0, R1, R2)	A0400800	R2=8	ADDC(R31, 7, R2)	C05F0007	R2=7
ADDC(R31, 5, R0)	C01F0005	R0=5	ST(R2, 0, R0)	64400000	MEM[Reg[R0]] ←Reg[R2]
ADDC(R31, -5, R1)	C03FFFFB	R1= −5	ADDC(R31,0,R0)	C01F0000	R0=0
CMPLEC(R0, 10, R30)	DBC0000A	R30=1	BNE(R0,0,R31)	7BE0FFE7	No jump
CMPLEC(R0, 5, R29)	DBA00005	R29=1	ADDC(R31,1,R1)	C03F0001	R1=1
CMPLEC(RO,2,R28)	DB800002	R28=0	BNE(R1,0,R31)	7BE1FFE5	Jump to first instruction
CMPLEC(R1, -6, R30)	DBC1FFFA	R30=0			

The first 5 lines of the first column are testing the `ADD` instruction.
The following 3 lines are testing the `AND` instruction. In the same instruction, we are testing every possibility (1&1, 1&0, 0&1, and 0&0).
The other lines of the first column and the first 2 lines of the second column are testing the `CMPLEC` instruction. The first 2 `ADDC` are used to put desired values inside the register file.
The following 7 lines are testing the `LD` instruction. We are using `R0` as a memory pointer. Then, we are storing 21 at address 0 in memory and loading the content at address 0 in `R1`. Finally, we are increasing `R0` by 4, storing 7 at address given by `R0`, and loading the content at the address given by `R0` in `R2`.
Finally, the remaining lines of the second column are testing the `BNE` instruction.

Logisim and ROMs

From time to time, Logisim may loose data of some ROMs. Our implementation of the β-machine is using 2 ROMs. Inside the instruction memory, the ROM is using addresses $\in [00; 2A]$. If some of these words are empty, please load the file `im_rom` inside this memory. Inside the control logic, the ROM is using addresses mentioned in the table in the section *Control logic*. If some of these words are empty, please load the file `cl_rom` inside the memory.

[1] Instructions associated to the lowest student id (20180521) of our group.
[2] See footnote 1.