Design of a Simple Processor

CS1800: Discrete Structures

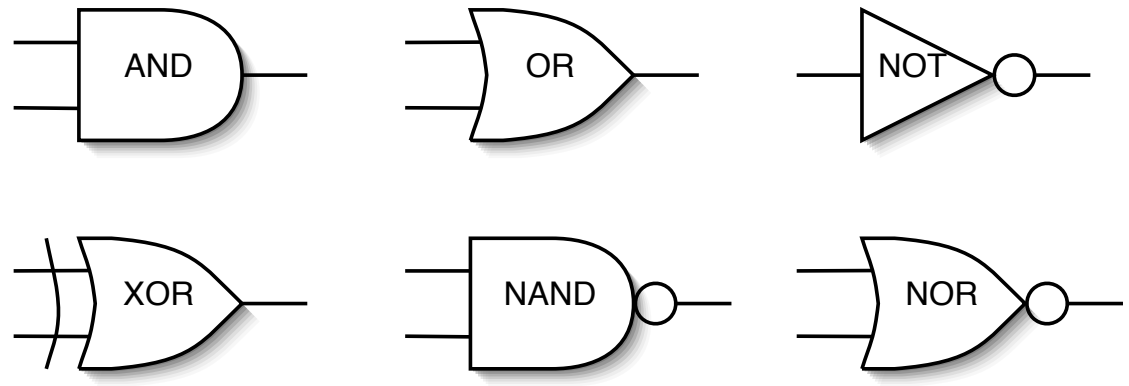
J. Aslam H. Fell R. Rajaraman R. Sundaram

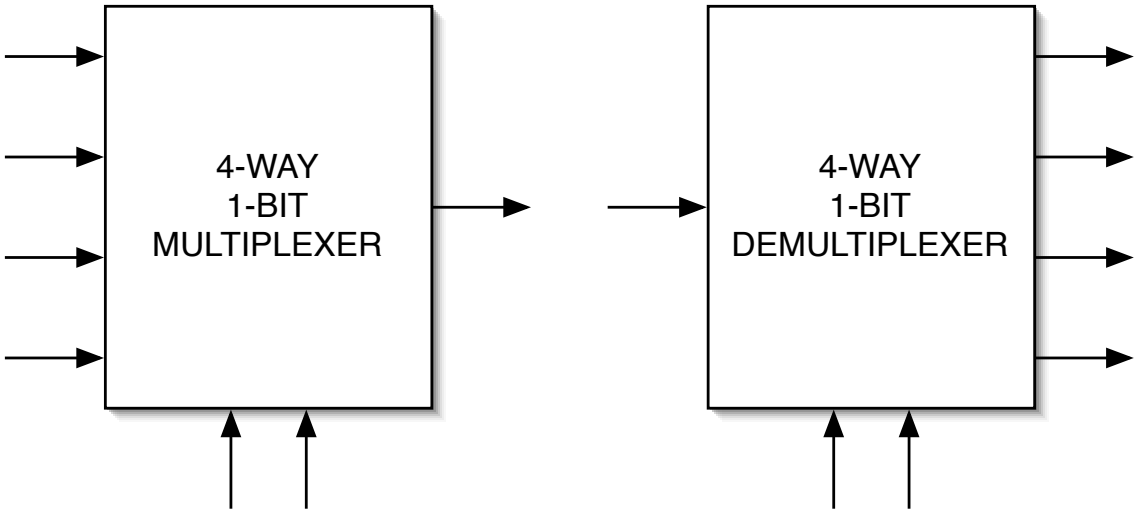
College of Computer & Information Science
Northeastern University

Based on material from CMU's Great Theoretical Ideas course

Outline

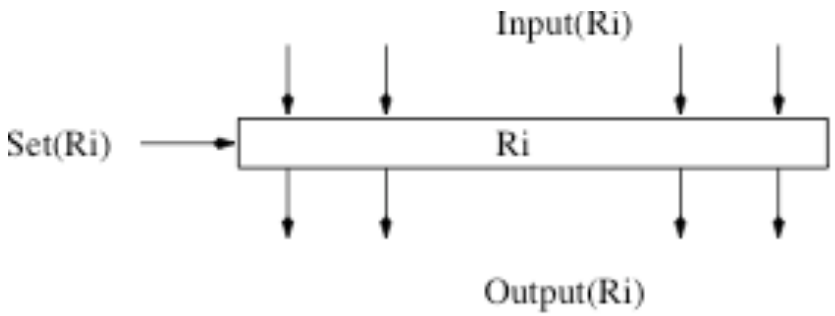
Building Blocks That We Have Already Seen





Clocked Gates and Registers

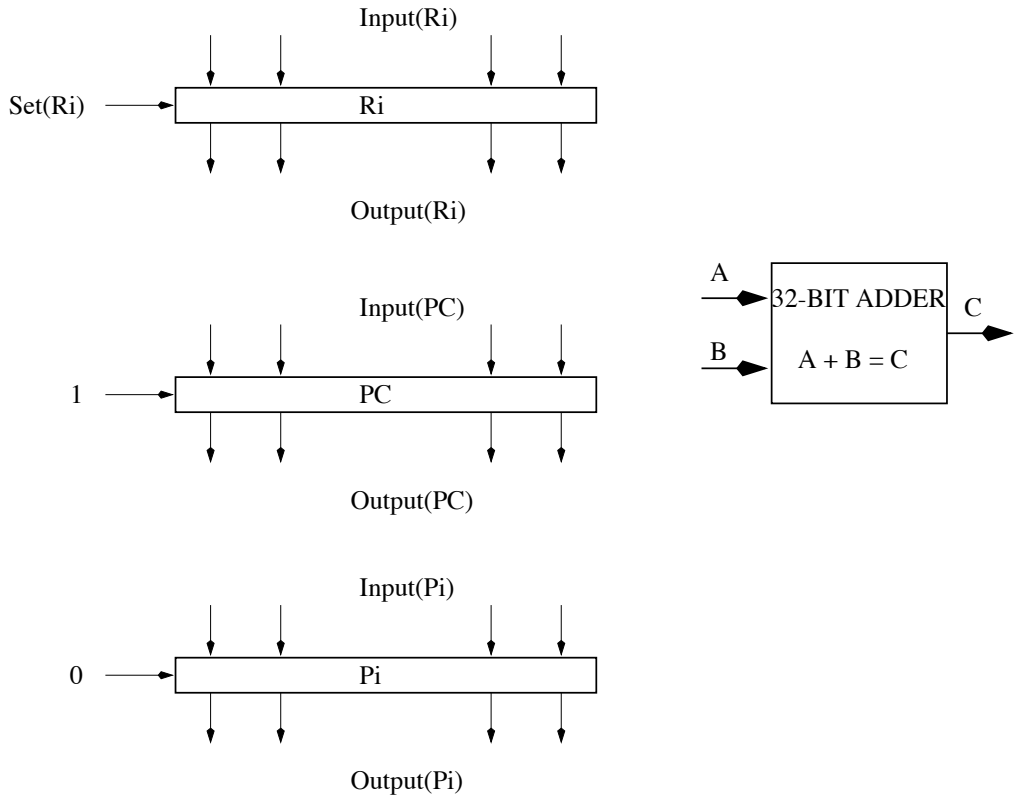
- Circuits using these basic gates are called *combinational circuits*.
  - No notion of time in computation: when inputs are fixed, outputs are permanently determined.
- We often need to regulate the flow the data; this is done using clocks.
- Using clocked gates, can build a register that stores values: if the set bit is 1 when the clock pulses, the output takes the input value, otherwise stays the same.



Registers and Adder

- Eight 32-bit registers $R_0, R_1, \ldots, R_7$ for holding data.
  - Useful to have the constant values 0 and 1 available.
  - Set register R_0 to hold 0 and R_1 to hold 1, permanently.
- 256 16-bit registers $P_0, P_1, \ldots, P_{255}$ for holding a program.
- An 8-bit register *PC* that will serve as a program counter.
- A 32-bit adder.

Registers and Adder, contd.



- Set bit for PC always 1 and set bit of program registers always 0.
- Set bits for the data registers determined by the instruction.

Instruction Set

- Four types of instructions:
  - *add*, *negate*, *load*, and *jump if zero*.
- A program is a sequence of instructions stored in the 256 program registers.
  - Each of these registers holds 16 bits.
  - The contents of the register specify the type of instruction and its operands.
  - Two bits (positions 14-15) specify the instruction.

The *add* Instruction

$$add\ R_a, R_b \rightarrow R_c$$

- Adds contents of registers R_a and R_b and stores result in register R_c.
- The indices a, b, and c are in bit positions 11-13, 8-10, and 5-7, respectively.
- Bit positions 0-4 will be ignored.
- Also increments the program counter by 1.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	a	a	a	b	b	b	c	c	c	0	0	0	0	0

The *negate* Instruction

$$neg\ R_a$$

- Replaces R_a with $-R_a$, using the two's complement representation.
- Index a is specified by bit positions 11-13.
- Also increments the program counter by 1.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	a	a	a	0	0	0	0	0	0	0	0	0	0	0

The *load* Instruction

$$lod\ d \rightarrow R_a$$

- Loads an 8-bit number d into the 8 low-order bit positions of register R_a.
- The index a is specified by bit positions 11-13.
- The value d is specified the 8 low-order bits (positions 0-7) of the instruction.
- Also increments the program counter by 1.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	a	a	a	0	0	0	d	d	d	d	d	d	d	d

The *jump if zero* Instruction

$$jiz\ R_a \rightarrow d$$

- If register R_a is 0, sets the program counter to the value specified by an 8-bit number d,
- The index a is specified by bit positions 11-13.
- The value d is specified in binary by the 8 low-order positions of the instruction; that is, positions 0-7.
- Otherwise the program counter PC is incremented by 1, as usual.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	a	a	a	0	0	0	d	d	d	d	d	d	d	d

Example Program

A program to add numbers from 1 to 10 and store the result in one of the registers

0	*lod*	$10 \rightarrow R_2$	10 010 000 00001010
1	*lod*	$0 \rightarrow R_3$	10 011 000 00000000
2	*lod*	$-1 \rightarrow R_4$	10 100 000 11111111
3	*add*	$R_2, R_3 \rightarrow R_3$	00 010 011 011 00000
4	*add*	$R_2, R_4 \rightarrow R_2$	00 010 100 010 00000
5	*jiz*	$R_2 \rightarrow 7$	11 010 000 00000111
6	*jiz*	$R_0 \rightarrow 3$	11 000 000 00000011
7	*jiz*	$R_0 \rightarrow 7$	11 000 000 00000111

Example Program

A program to add numbers from 1 to 10 and store the result in one of the registers

0	*lod*	$10 \rightarrow R_2$	10 010 000 00001010
1	*lod*	$0 \rightarrow R_3$	10 011 000 00000000
2	*lod*	$-1 \rightarrow R_4$	10 100 000 11111111
3	*add*	$R_2, R_3 \rightarrow R_3$	00 010 011 011 00000
4	*add*	$R_2, R_4 \rightarrow R_2$	00 010 100 010 00000
5	*jiz*	$R_2 \rightarrow 7$	11 010 000 00000111
6	*jiz*	$R_0 \rightarrow 3$	11 000 000 00000011
7	*jiz*	$R_0 \rightarrow 7$	11 000 000 00000111

Example Program

A program to add numbers from 1 to 10 and store the result in one of the registers

0	*lod*	$10 \rightarrow R_2$	10 010 000 00001010
1	*lod*	$0 \rightarrow R_3$	10 011 000 00000000
2	*lod*	$-1 \rightarrow R_4$	10 100 000 11111111
3	*add*	$R_2, R_3 \rightarrow R_3$	00 010 011 011 00000
4	*add*	$R_2, R_4 \rightarrow R_2$	00 010 100 010 00000
5	*jiz*	$R_2 \rightarrow 7$	11 010 000 00000111
6	*jiz*	$R_0 \rightarrow 3$	11 000 000 00000011
7	*jiz*	$R_0 \rightarrow 7$	11 000 000 00000111

Example Program

A program to add numbers from 1 to 10 and store the result in one of the registers

0	*lod*	$10 \rightarrow R_2$	10 010 000 00001010
1	*lod*	$0 \rightarrow R_3$	10 011 000 00000000
2	*lod*	$-1 \rightarrow R_4$	10 100 000 11111111
3	*add*	$R_2, R_3 \rightarrow R_3$	00 010 011 011 00000
4	*add*	$R_2, R_4 \rightarrow R_2$	00 010 100 010 00000
5	*jiz*	$R_2 \rightarrow 7$	11 010 000 00000111
6	*jiz*	$R_0 \rightarrow 3$	11 000 000 00000011
7	*jiz*	$R_0 \rightarrow 7$	11 000 000 00000111

Example Program

A program to add numbers from 1 to 10 and store the result in one of the registers

0	*lod*	$10 \rightarrow R_2$	10 010 000 00001010
1	*lod*	$0 \rightarrow R_3$	10 011 000 00000000
2	*lod*	$-1 \rightarrow R_4$	10 100 000 11111111
3	*add*	$R_2, R_3 \rightarrow R_3$	00 010 011 011 00000
4	*add*	$R_2, R_4 \rightarrow R_2$	00 010 100 010 00000
5	*jiz*	$R_2 \rightarrow 7$	11 010 000 00000111
6	*jiz*	$R_0 \rightarrow 3$	11 000 000 00000011
7	*jiz*	$R_0 \rightarrow 7$	11 000 000 00000111

Example Program

A program to add numbers from 1 to 10 and store the result in one of the registers

0	*lod*	$10 \rightarrow R_2$	10 010 000 00001010
1	*lod*	$0 \rightarrow R_3$	10 011 000 00000000
2	*lod*	$-1 \rightarrow R_4$	10 100 000 11111111
3	*add*	$R_2, R_3 \rightarrow R_3$	00 010 011 011 00000
4	*add*	$R_2, R_4 \rightarrow R_2$	00 010 100 010 00000
5	*jiz*	$R_2 \rightarrow 7$	11 010 000 00000111
6	*jiz*	$R_0 \rightarrow 3$	11 000 000 00000011
7	*jiz*	$R_0 \rightarrow 7$	11 000 000 00000111

Example Program

A program to add numbers from 1 to 10 and store the result in one of the registers

0	*lod*	$10 \rightarrow R_2$	10 010 000 00001010
1	*lod*	$0 \rightarrow R_3$	10 011 000 00000000
2	*lod*	$-1 \rightarrow R_4$	10 100 000 11111111
3	*add*	$R_2, R_3 \rightarrow R_3$	00 010 011 011 00000
4	*add*	$R_2, R_4 \rightarrow R_2$	00 010 100 010 00000
5	*jiz*	$R_2 \rightarrow 7$	11 010 000 00000111
6	*jiz*	$R_0 \rightarrow 3$	11 000 000 00000011
7	*jiz*	$R_0 \rightarrow 7$	11 000 000 00000111

Example Program

A program to add numbers from 1 to 10 and store the result in one of the registers

0	*lod*	$10 \rightarrow R_2$	10 010 000 00001010
1	*lod*	$0 \rightarrow R_3$	10 011 000 00000000
2	*lod*	$-1 \rightarrow R_4$	10 100 000 11111111
3	*add*	$R_2, R_3 \rightarrow R_3$	00 010 011 011 00000
4	*add*	$R_2, R_4 \rightarrow R_2$	00 010 100 010 00000
5	*jiz*	$R_2 \rightarrow 7$	11 010 000 00000111
6	*jiz*	$R_0 \rightarrow 3$	11 000 000 00000011
7	*jiz*	$R_0 \rightarrow 7$	11 000 000 00000111

Example Program

A program to add numbers from 1 to 10 and store the result in one of the registers

0	*lod*	$10 \rightarrow R_2$	10 010 000 00001010
1	*lod*	$0 \rightarrow R_3$	10 011 000 00000000
2	*lod*	$-1 \rightarrow R_4$	10 100 000 11111111
3	*add*	$R_2, R_3 \rightarrow R_3$	00 010 011 011 00000
4	*add*	$R_2, R_4 \rightarrow R_2$	00 010 100 010 00000
5	*jiz*	$R_2 \rightarrow 7$	11 010 000 00000111
6	*jiz*	$R_0 \rightarrow 3$	11 000 000 00000011
7	*jiz*	$R_0 \rightarrow 7$	11 000 000 00000111

Example Program

A program to add numbers from 1 to 10 and store the result in one of the registers

0	*lod*	$10 \rightarrow R_2$	10 010 000 00001010
1	*lod*	$0 \rightarrow R_3$	10 011 000 00000000
2	*lod*	$-1 \rightarrow R_4$	10 100 000 11111111
3	*add*	$R_2, R_3 \rightarrow R_3$	00 010 011 011 00000
4	*add*	$R_2, R_4 \rightarrow R_2$	00 010 100 010 00000
5	*jiz*	$R_2 \rightarrow 7$	11 010 000 00000111
6	*jiz*	$R_0 \rightarrow 3$	11 000 000 00000011
7	*jiz*	$R_0 \rightarrow 7$	11 000 000 00000111

Example Program

A program to add numbers from 1 to 10 and store the result in one of the registers

0	*lod*	$10 \rightarrow R_2$	10 010 000 00001010
1	*lod*	$0 \rightarrow R_3$	10 011 000 00000000
2	*lod*	$-1 \rightarrow R_4$	10 100 000 11111111
3	*add*	$R_2, R_3 \rightarrow R_3$	00 010 011 011 00000
4	*add*	$R_2, R_4 \rightarrow R_2$	00 010 100 010 00000
5	*jiz*	$R_2 \rightarrow 7$	11 010 000 00000111
6	*jiz*	$R_0 \rightarrow 3$	11 000 000 00000011
7	*jiz*	$R_0 \rightarrow 7$	11 000 000 00000111

Example Program

A program to add numbers from 1 to 10 and store the result in one of the registers

0	*lod*	$10 \rightarrow R_2$	10 010 000 00001010
1	*lod*	$0 \rightarrow R_3$	10 011 000 00000000
2	*lod*	$-1 \rightarrow R_4$	10 100 000 11111111
3	*add*	$R_2, R_3 \rightarrow R_3$	00 010 011 011 00000
4	*add*	$R_2, R_4 \rightarrow R_2$	00 010 100 010 00000
5	*jiz*	$R_2 \rightarrow 7$	11 010 000 00000111
6	*jiz*	$R_0 \rightarrow 3$	11 000 000 00000011
7	*jiz*	$R_0 \rightarrow 7$	11 000 000 00000111

Example Program

A program to add numbers from 1 to 10 and store the result in one of the registers

0	*lod*	$10 \rightarrow R_2$	10 010 000 00001010
1	*lod*	$0 \rightarrow R_3$	10 011 000 00000000
2	*lod*	$-1 \rightarrow R_4$	10 100 000 11111111
3	*add*	$R_2, R_3 \rightarrow R_3$	00 010 011 011 00000
4	*add*	$R_2, R_4 \rightarrow R_2$	00 010 100 010 00000
5	*jiz*	$R_2 \rightarrow 7$	11 010 000 00000111
6	*jiz*	$R_0 \rightarrow 3$	11 000 000 00000011
7	*jiz*	$R_0 \rightarrow 7$	11 000 000 00000111

Example Program

A program to add numbers from 1 to 10 and store the result in one of the registers

0	*lod*	$10 \rightarrow R_2$	10 010 000 00001010
1	*lod*	$0 \rightarrow R_3$	10 011 000 00000000
2	*lod*	$-1 \rightarrow R_4$	10 100 000 11111111
3	*add*	$R_2, R_3 \rightarrow R_3$	00 010 011 011 00000
4	*add*	$R_2, R_4 \rightarrow R_2$	00 010 100 010 00000
5	*jiz*	$R_2 \rightarrow 7$	11 010 000 00000111
6	*jiz*	$R_0 \rightarrow 3$	11 000 000 00000011
7	*jiz*	$R_0 \rightarrow 7$	11 000 000 00000111

Example Program

A program to add numbers from 1 to 10 and store the result in one of the registers

0	*lod*	$10 \rightarrow R_2$	10 010 000 00001010
1	*lod*	$0 \rightarrow R_3$	10 011 000 00000000
2	*lod*	$-1 \rightarrow R_4$	10 100 000 11111111
3	*add*	$R_2, R_3 \rightarrow R_3$	00 010 011 011 00000
4	*add*	$R_2, R_4 \rightarrow R_2$	00 010 100 010 00000
5	*jiz*	$R_2 \rightarrow 7$	11 010 000 00000111
6	*jiz*	$R_0 \rightarrow 3$	11 000 000 00000011
7	*jiz*	$R_0 \rightarrow 7$	11 000 000 00000111

Example Program

A program to add numbers from 1 to 10 and store the result in one of the registers

0	*lod*	$10 \rightarrow R_2$	10 010 000 00001010
1	*lod*	$0 \rightarrow R_3$	10 011 000 00000000
2	*lod*	$-1 \rightarrow R_4$	10 100 000 11111111
3	*add*	$R_2, R_3 \rightarrow R_3$	00 010 011 011 00000
4	*add*	$R_2, R_4 \rightarrow R_2$	00 010 100 010 00000
5	*jiz*	$R_2 \rightarrow 7$	11 010 000 00000111
6	*jiz*	$R_0 \rightarrow 3$	11 000 000 00000011
7	*jiz*	$R_0 \rightarrow 7$	11 000 000 00000111

Example Program

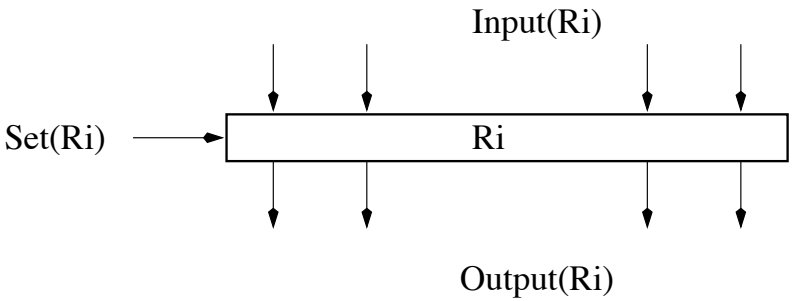
A program to add numbers from 1 to 10 and store the result in one of the registers

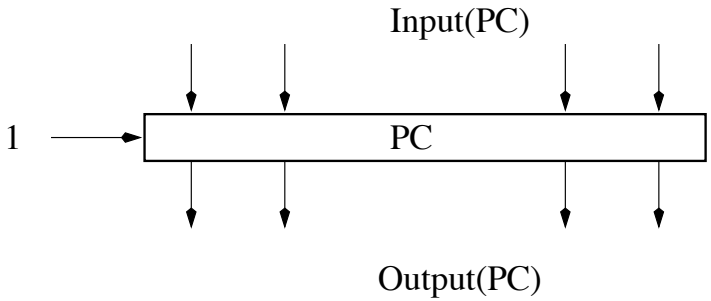
0	*lod*	$10 \rightarrow R_2$	10 010 000 00001010
1	*lod*	$0 \rightarrow R_3$	10 011 000 00000000
2	*lod*	$-1 \rightarrow R_4$	10 100 000 11111111
3	*add*	$R_2, R_3 \rightarrow R_3$	00 010 011 011 00000
4	*add*	$R_2, R_4 \rightarrow R_2$	00 010 100 010 00000
5	*jiz*	$R_2 \rightarrow 7$	11 010 000 00000111
6	*jiz*	$R_0 \rightarrow 3$	11 000 000 00000011
7	*jiz*	$R_0 \rightarrow 7$	11 000 000 00000111

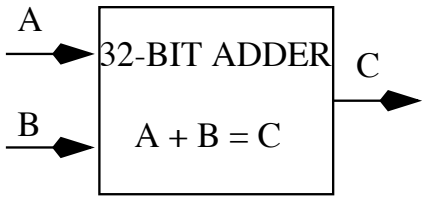
Design of the CPU

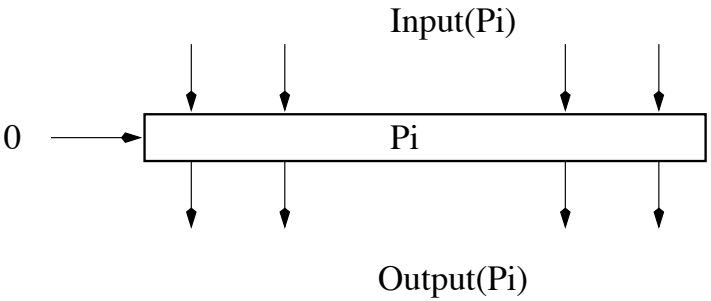
- Lay out the registers, giving names to the input bits, output bits, and registers (have already done this).
- Design the circuits for extracting the next instruction and incrementing the PC.
- Design the circuits for implementing the add, negate, load, and jump-if-zero operations.
- Design the circuit that determines the input and the set bits for the data registers and the PC.
- These circuits can be put together by matching the appropriate labeled inputs and outputs.

Registers and Adder

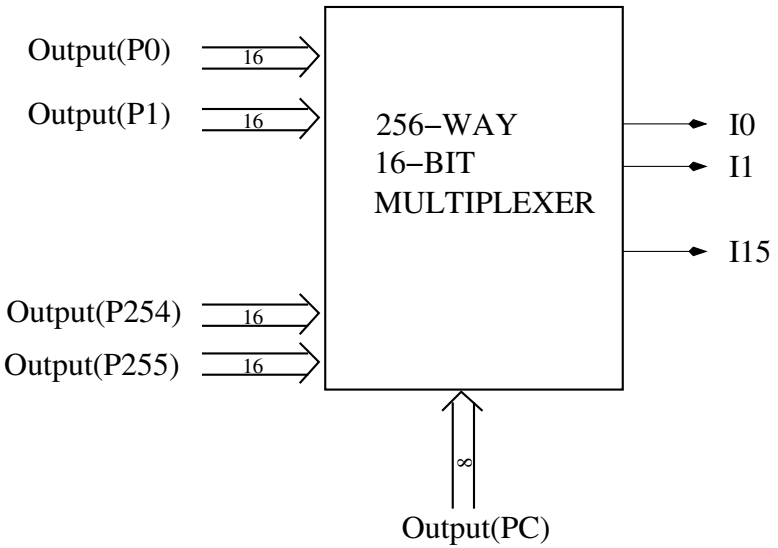


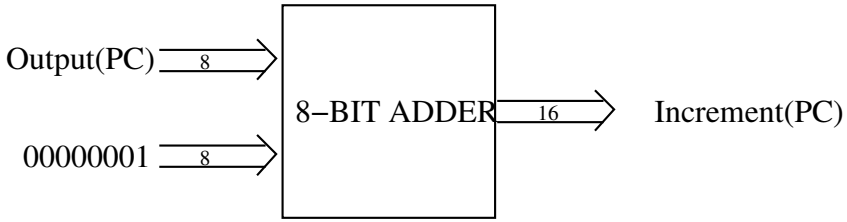


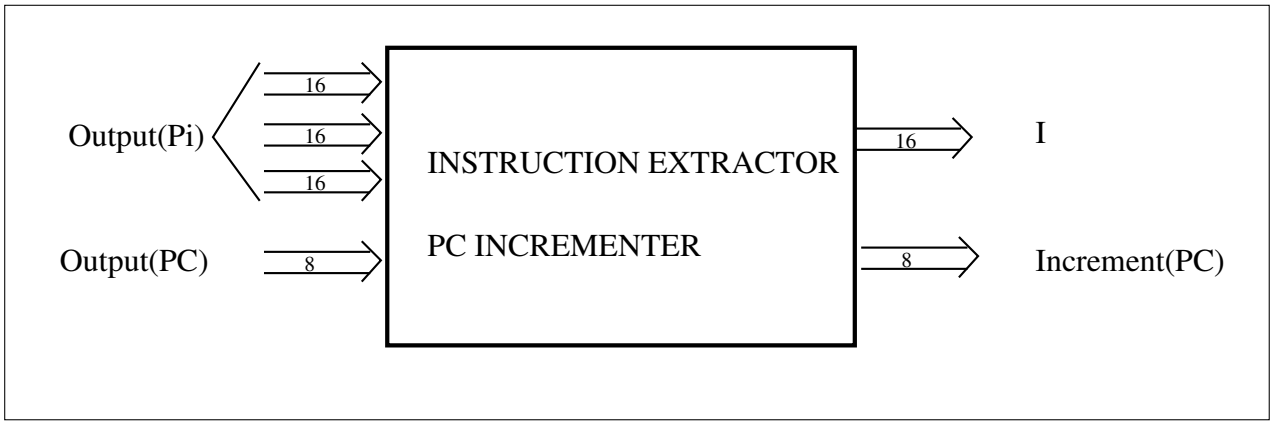




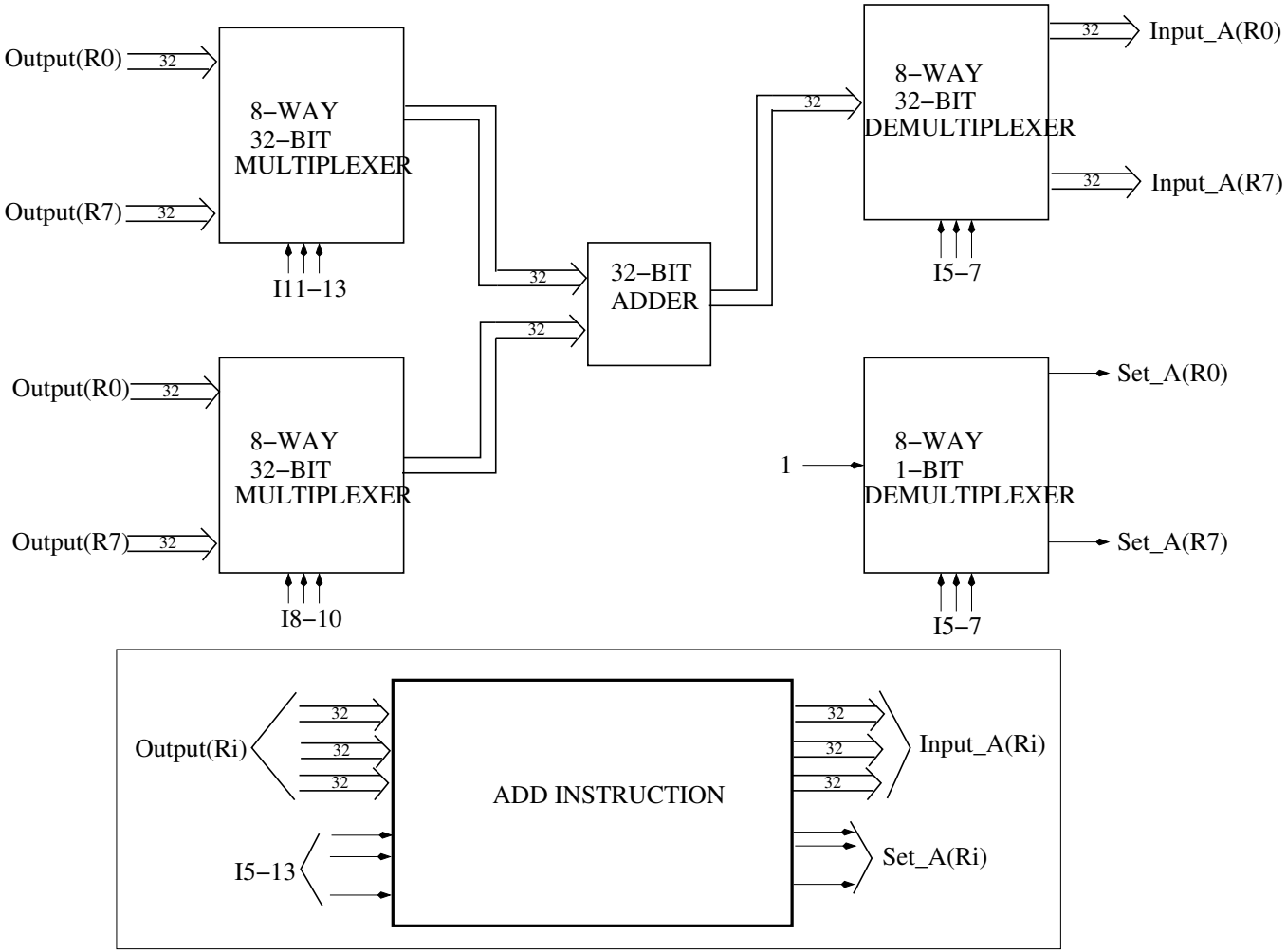
Extracting the instruction and incrementing the PC



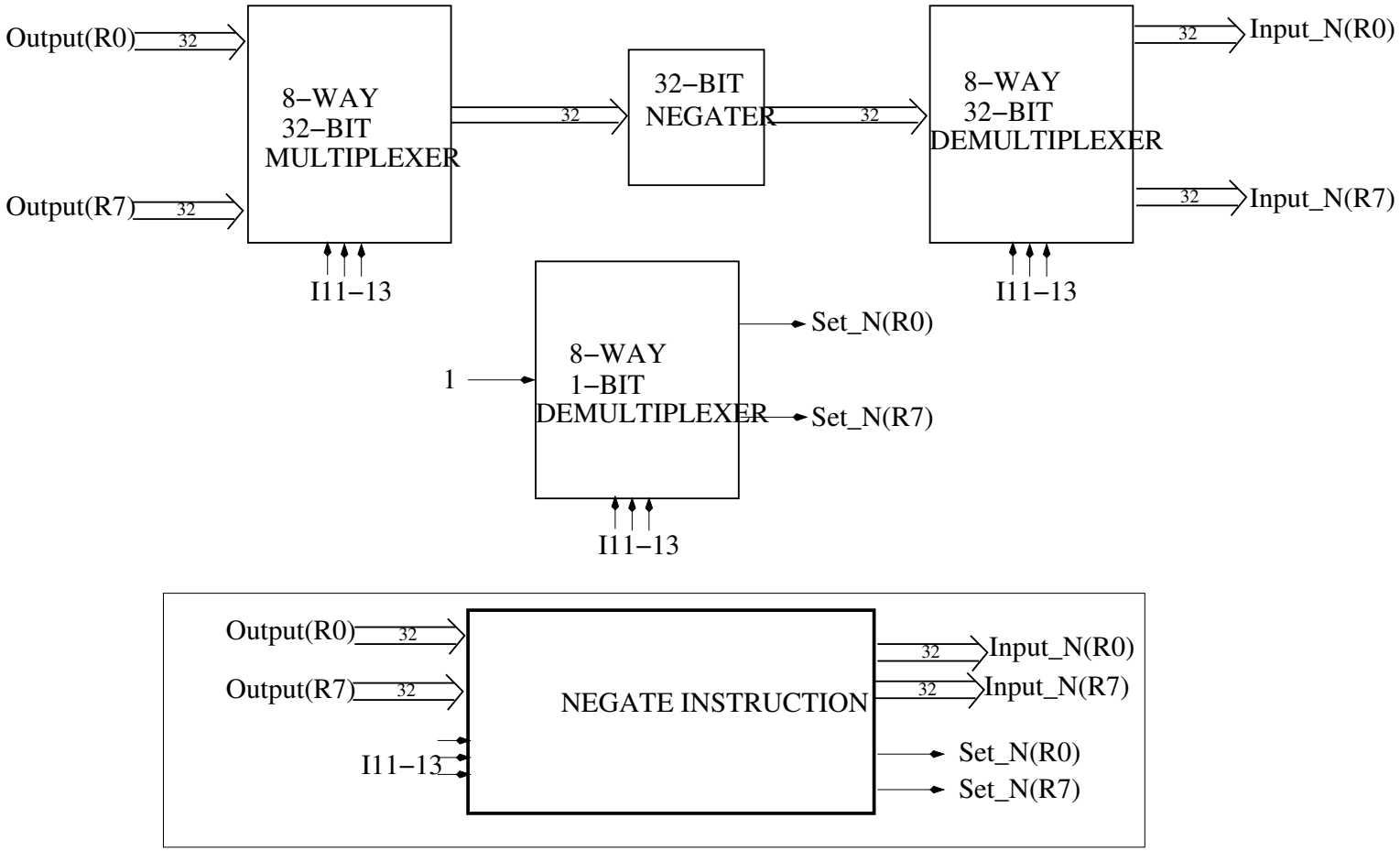




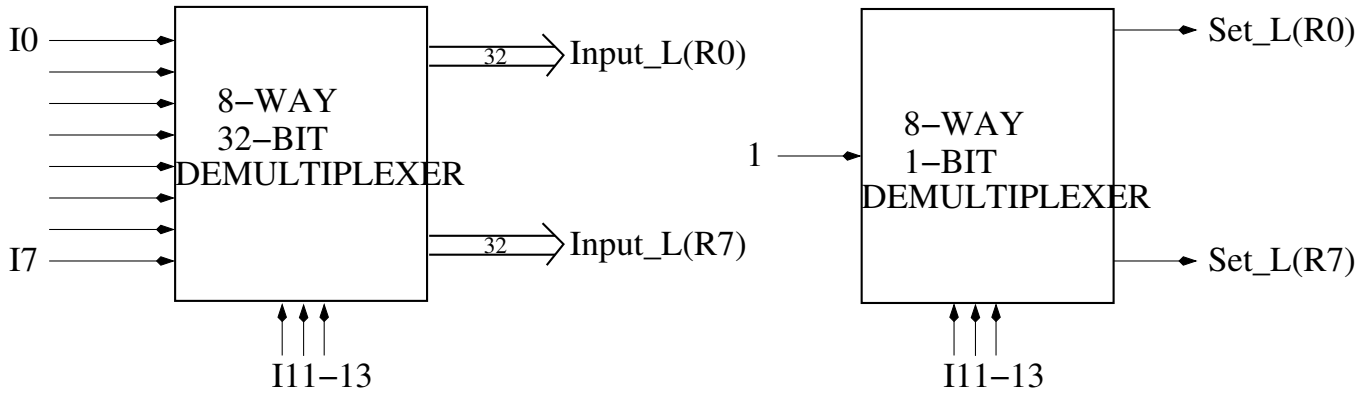
Circuit for the add instruction

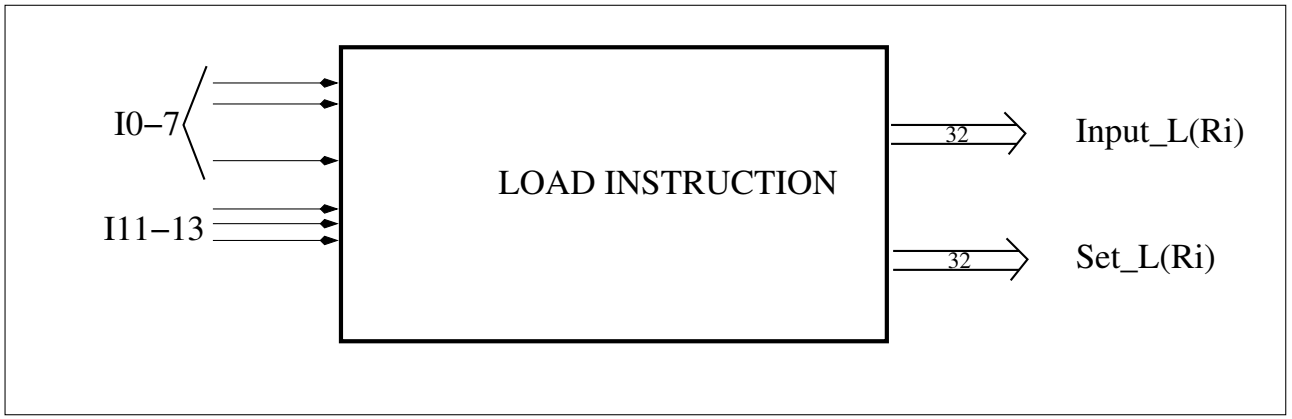


Circuit for the negate instruction

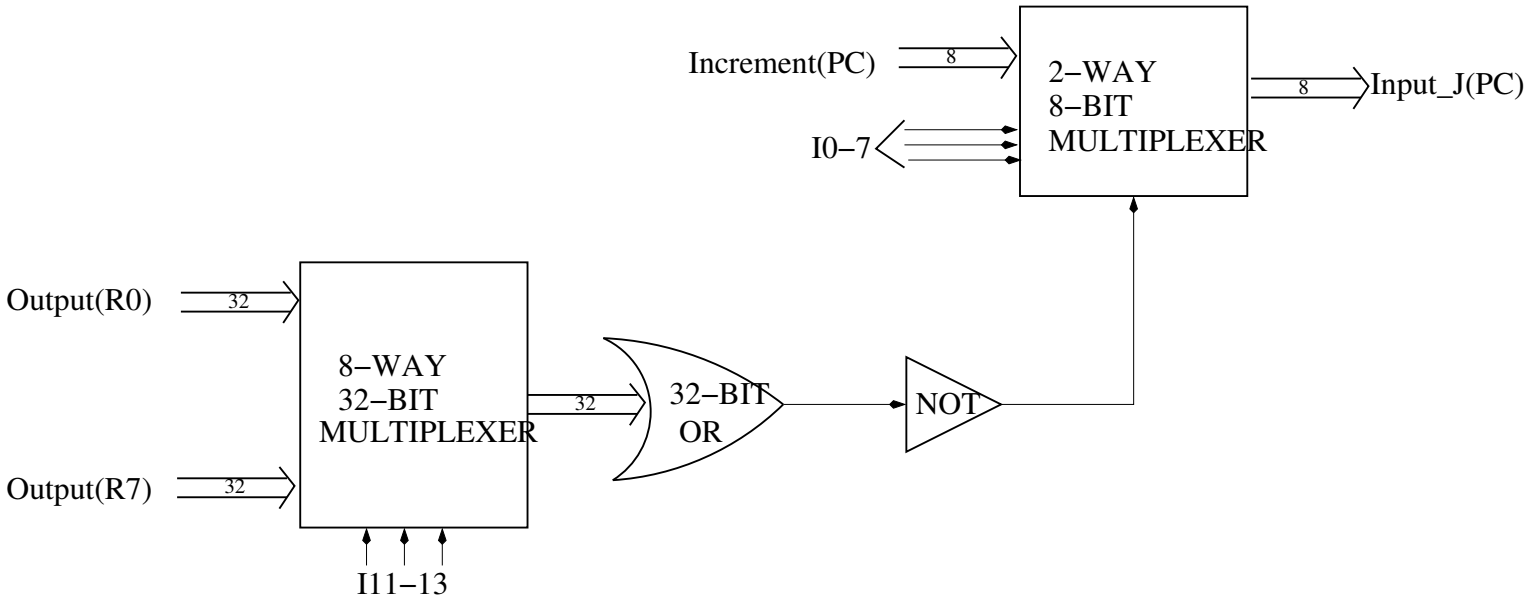


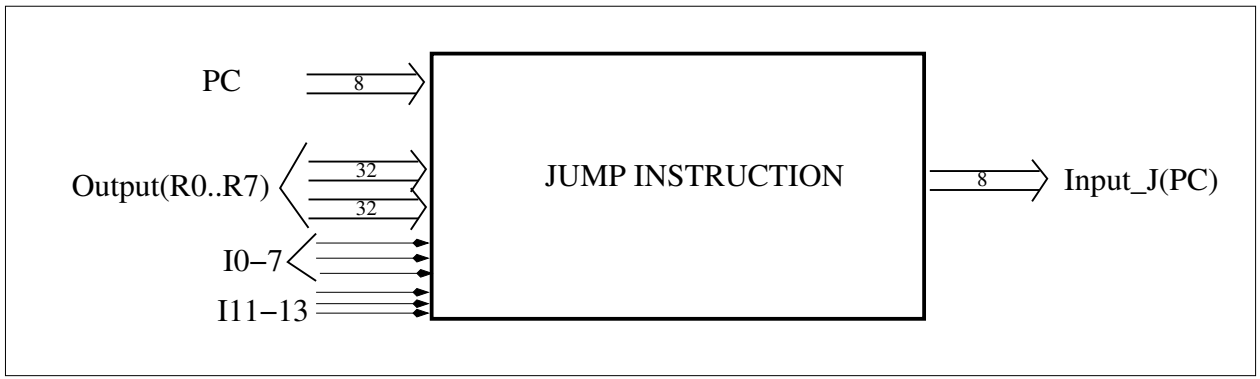
Circuit for the load instruction





Circuit for the jump-if-zero instruction





Storing the new values into the data registers and PC

- The preceding circuits give us potential values to be stored in the data registers.
- The new values for the data registers are determined by instruction type.