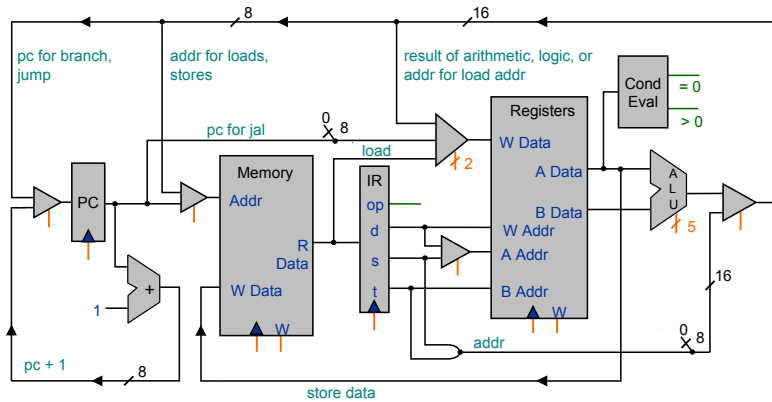


6.5: TOY Machine Architecture



Introduction to Computer Science · Robert Sedgewick and Kevin Wayne · <http://www.cs.Princeton.EDU/IntroCS>

The TOY Machine

TOY machine.

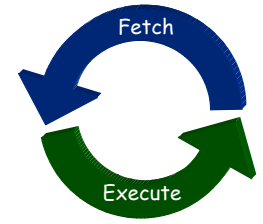
- 256 16-bit words of memory.
- 16 16-bit registers.
- 1 8-bit program counter.
- 16 instructions types.

What we've done.

- Written programs for the TOY machine.
- Software implementation of fetch-execute cycle.
 - TOY simulator.

Our goal today.

- Hardware implementation of fetch-execute cycle.
 - TOY computer.



Designing a Processor

How to build a microprocessor?

- ➔ • Develop instruction set architecture (ISA).
 - 16-bit words, 16 TOY machine instructions
- Determine major components.
 - ALU, memory, registers, program counter
- Determine datapath requirements.
 - "flow" of bits
- Establish clocking methodology.
 - 2-cycle design: fetch, execute
- Analyze how to implement each instruction.
 - determine settings of control signals

Instruction Set Architecture

Instruction set architecture (ISA).

- 16-bit words, 256 words of memory, 16 registers.
- Determine set of primitive instructions.
 - too narrow => cumbersome to program
 - too broad => cumbersome to build hardware
- TOY machine: 16 instructions.

Instructions	
0:	halt
1:	add
2:	subtract
3:	and
4:	xor
5:	shift left
6:	shift right
7:	load address

Instructions	
8:	load
9:	store
A:	load indirect
B:	store indirect
C:	branch zero
D:	branch positive
E:	jump register
F:	jump and link

Designing a Processor

How to build a microprocessor?

- Develop instruction set architecture (ISA).
 - 16-bit words, 16 TOY machine instructions
- ➔ • Determine major components.
 - ALU, memory, registers, program counter
- Determine datapath requirements.
 - "flow" of bits
- Establish clocking methodology.
 - 2-cycle design: fetch, execute
- Analyze how to implement each instruction.
 - determine settings of control signals

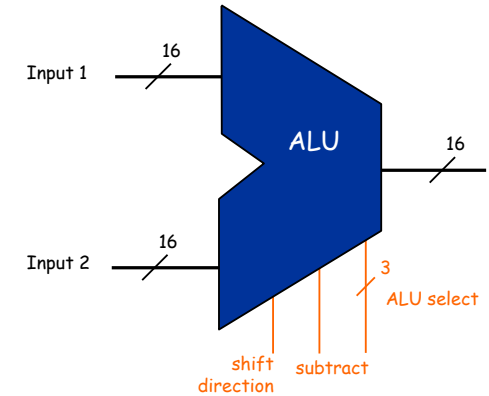
5

Arithmetic Logic Unit

TOY ALU.

- Big combinational circuit.
- 16-bit bus.
- Add, subtract, and, xor, shift left, shift right, copy input 2.

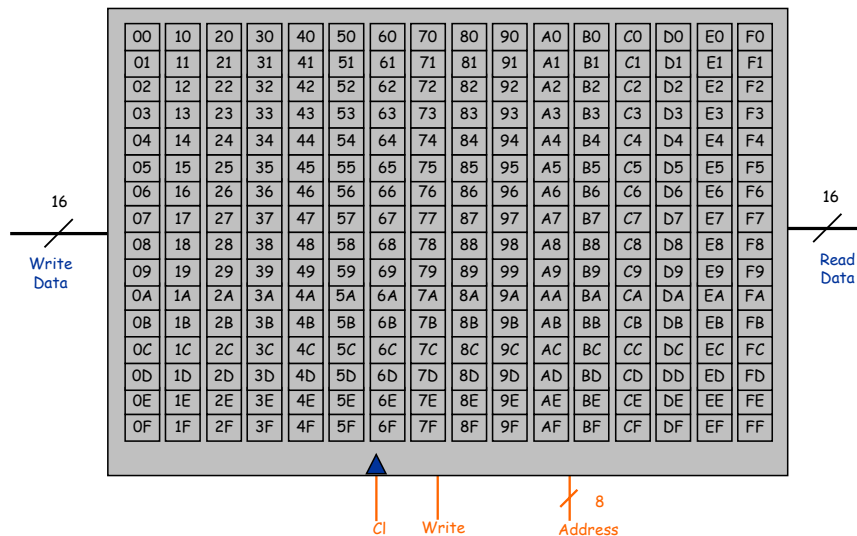
op	2	1	0
+, -	0	0	0
&	0	0	1
^	0	1	0
<<, >>	0	1	1
input 2	1	0	0



6

Main Memory

TOY main memory: 256 x 16-bit register file.

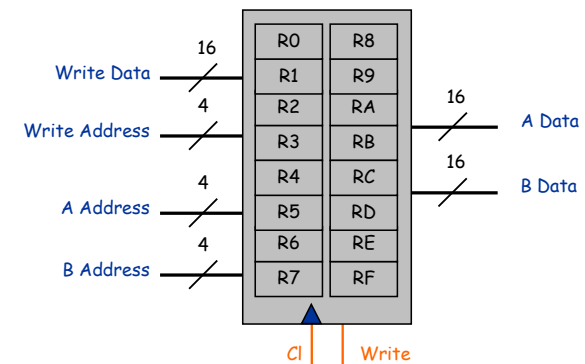


7

Registers

TOY registers: fancy 16 x 16-bit register file.

- Want to be able to read two registers, and write to a third in the same instructions: $R1 \leftarrow R2 + R3$.
- 3 address inputs, 2 data outputs.
- Add extra bank of muxes for a second read port.



8

Designing a Processor

How to build a microprocessor?

- Develop instruction set architecture (ISA).
 - 16-bit words, 16 TOY machine instructions
- Determine major components.
 - ALU, memory, registers, program counter
- ➔ • Determine datapath requirements.
 - "flow" of bits
- Establish clocking methodology.
 - 2-cycle design: fetch, execute
- Analyze how to implement each instruction.
 - determine settings of control signals

9

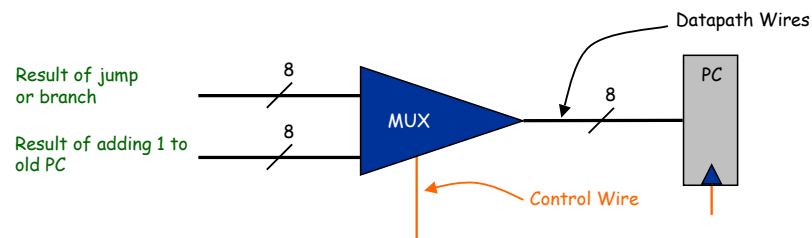
Datapath and Control

Datapath.

- Layout and interconnection of components.
- Must accommodate all instruction types.

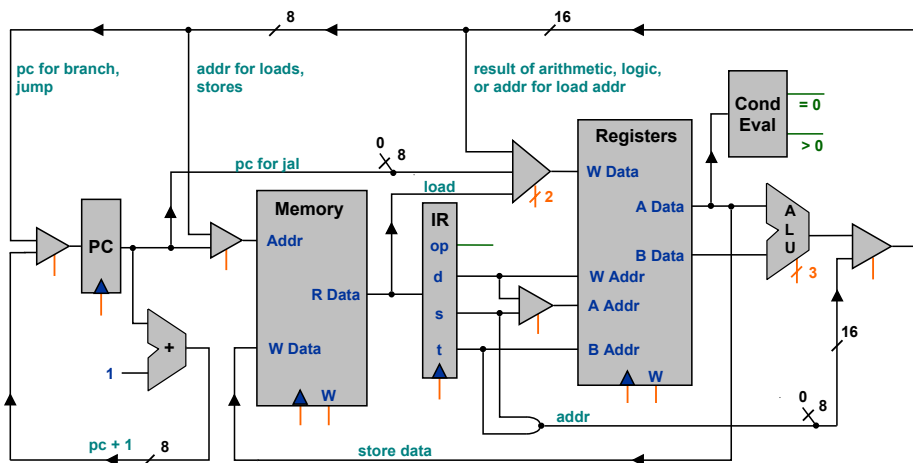
Control.

- Choreographs the "flow" of information on the datapath.
- Depending on instruction, different control wires are turned on.



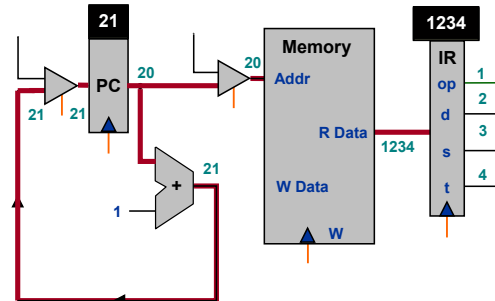
10

The TOY Datapath



13

The TOY Datapath: Add



Before fetch:

pc = 20, mem[20] = 1234

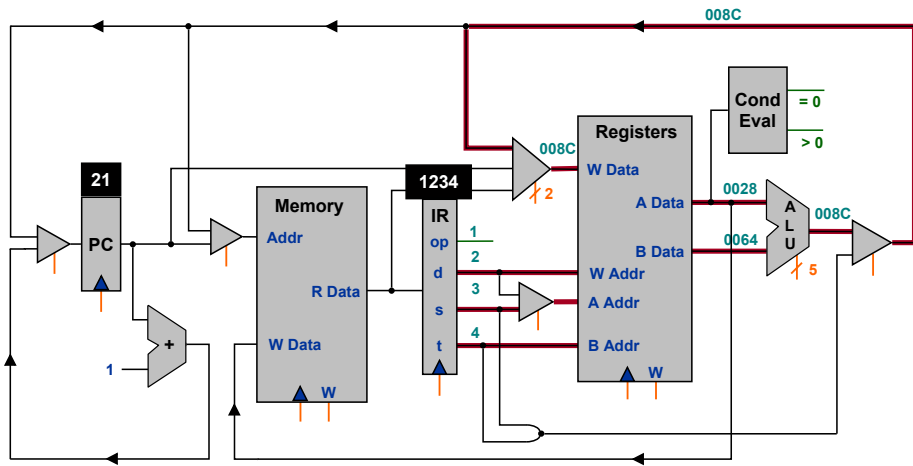
After fetch:

pc = 21

IR = 1234: R[2] ← R[3] + R[4]

14

The TOY Datapath: Add



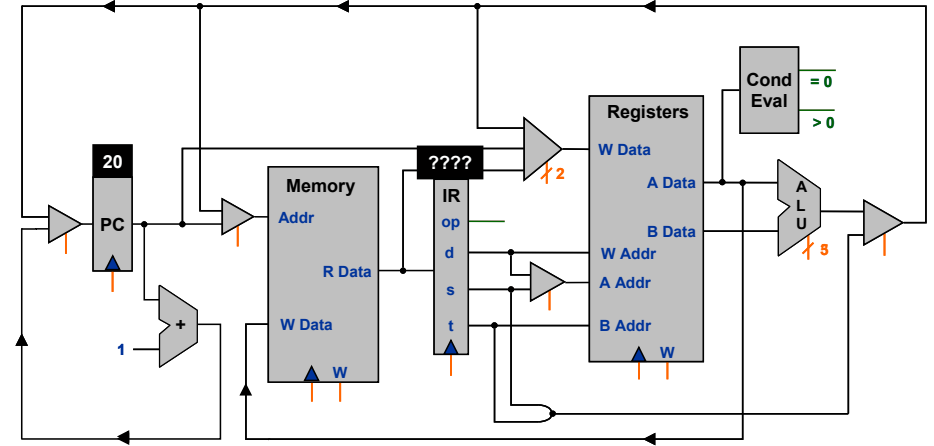
Before execute:

pc = 21
 IR = 1234: R[2] ← R[3] + R[4]
 R[3] = 0028, R[4] = 0064

After execute:

pc = 21
 R[2] = 008C

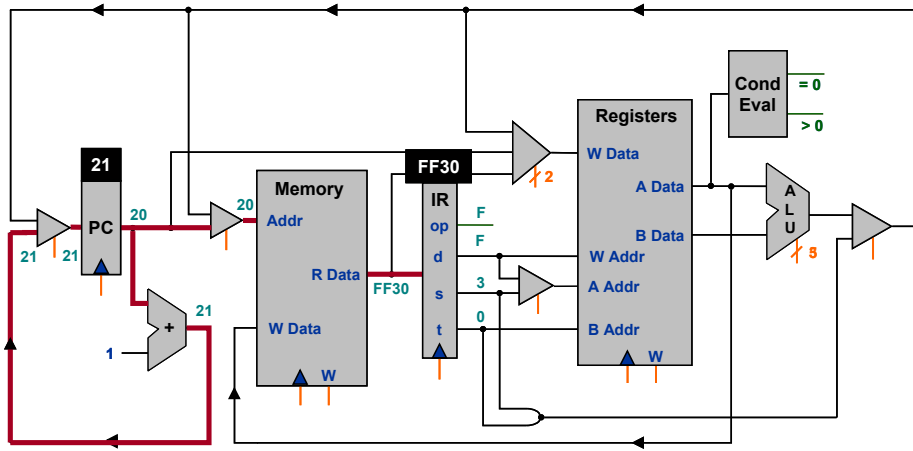
The TOY Datapath: Jump and Link



Before fetch:

pc = 20
 mem[20] = FF30

The TOY Datapath: Jump and Link



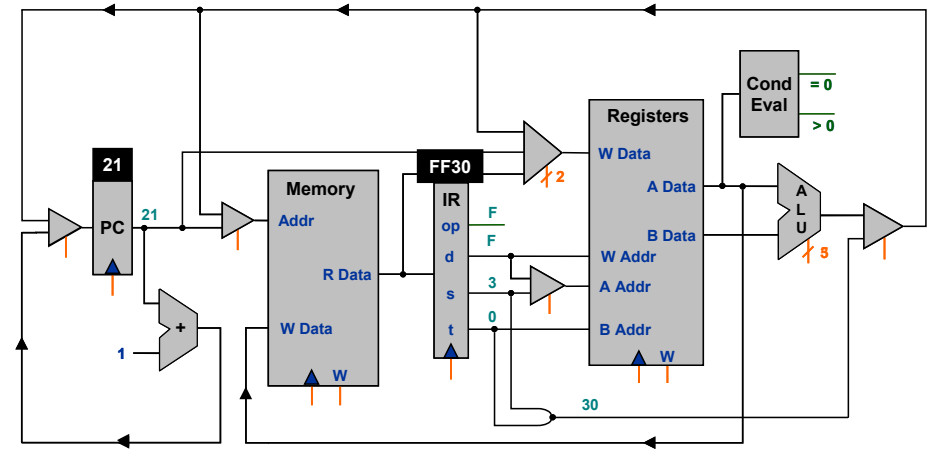
Before fetch:

pc = 20
 mem[20] = FF30

After fetch:

pc = 21
 IR = FF30: R[F] ← 21; pc ← 30

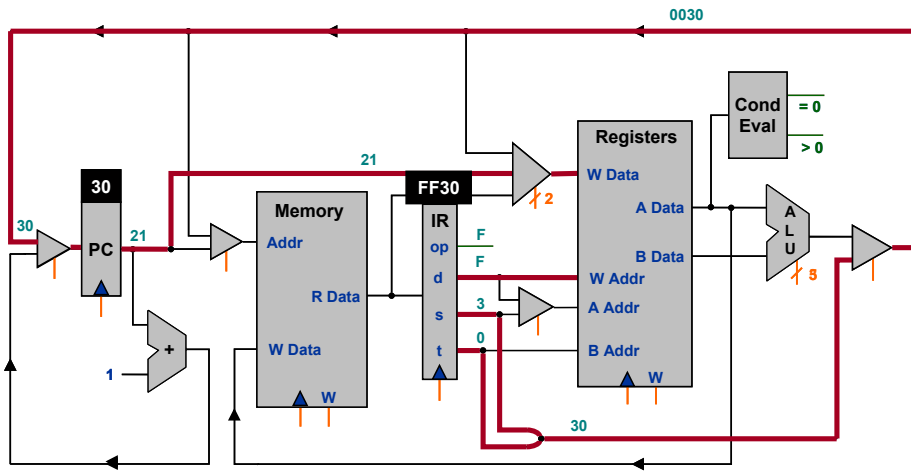
The TOY Datapath: Jump and Link



Before execute:

pc = 21
 IR = FF30: R[F] ← 21; pc ← 30

The TOY Datapath: Jump and Link



Before execute:

pc = 21

IR = FF30: R[F] ← 21; pc ← 30

After execute:

pc = 30

R[F] = 21

19

Designing a Processor

How to build a microprocessor?

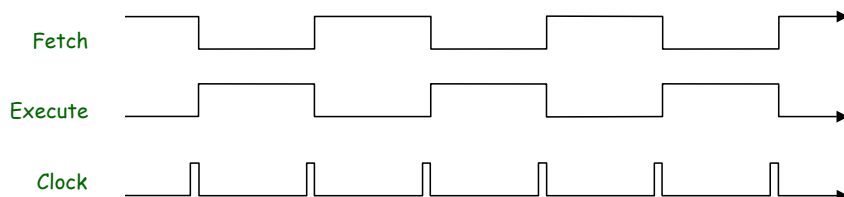
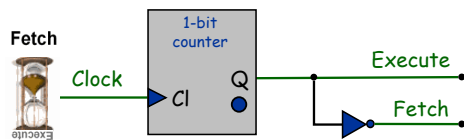
- Develop instruction set architecture (ISA).
 - 16-bit words, 16 TOY machine instructions
- Determine major components.
 - ALU, memory, registers, program counter
- Determine datapath requirements.
 - "flow" of bits
- ➔ • Establish clocking methodology.
 - 2-cycle design: fetch, execute
- Analyze how to implement each instruction.
 - determine settings of control signals

21

Clocking Methodology

Two cycle design (fetch and execute).

- Use 1-bit counter to distinguish between 2 cycles.
- Use two cycles since fetch and execute phases each access memory and alter program counter.



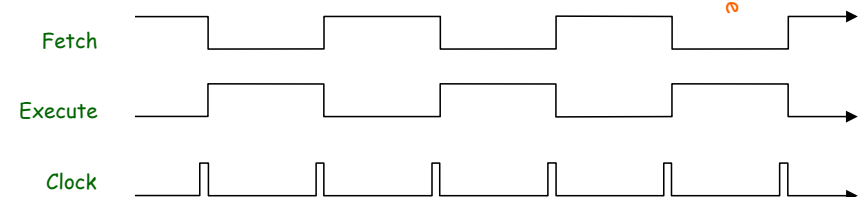
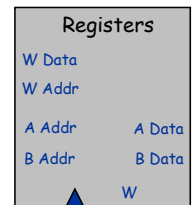
22

Clocking Methodology

4 distinguishable events.

- During fetch phase.
- During execute phase.
- At very end of execute phase.
- At very end of fetch phase.

can only write to register at very end of execute phase



23

Designing a Processor

How to build a microprocessor?

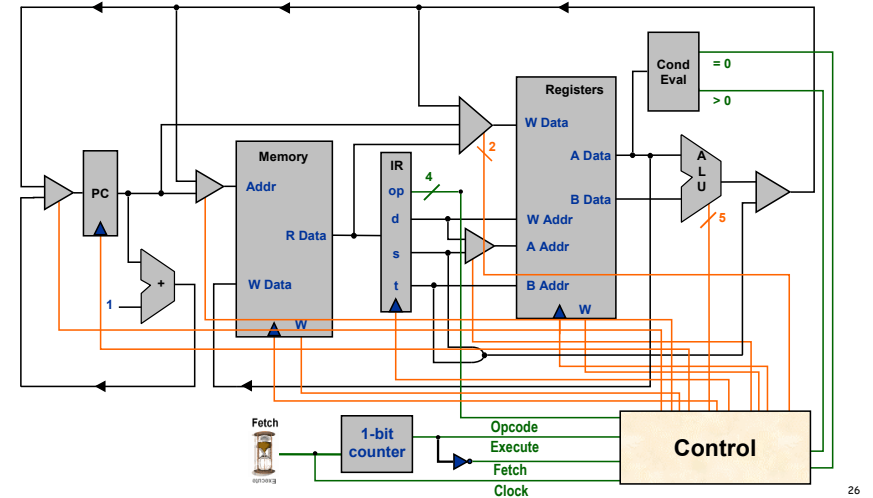
- Develop instruction set architecture (ISA).
 - 16-bit words, 16 TOY machine instructions
- Determine major components.
 - ALU, memory, registers, program counter
- Determine datapath requirements.
 - "flow" of bits
- Establish clocking methodology.
 - 2-cycle design: fetch, execute
- ➔ • Analyze how to implement each instruction.
 - determine settings of control signals

24

Control

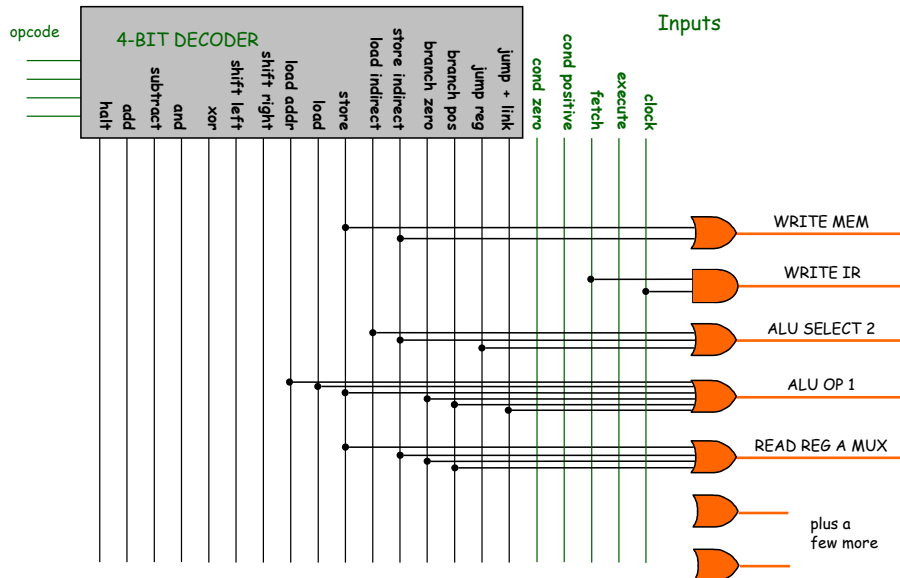
Control: controls components, enables connections.

- Input: opcode, clock, conditional evaluation. (green)
- Output: control wires. (orange)



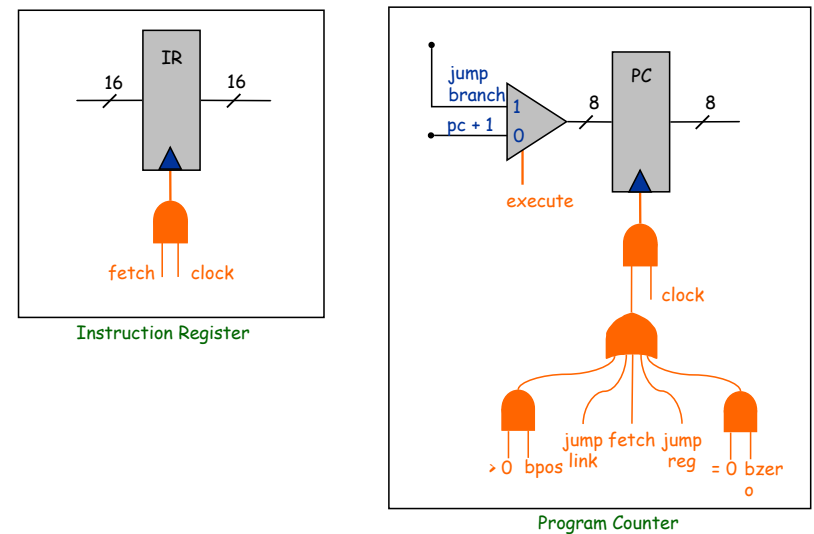
26

Implementation of Control



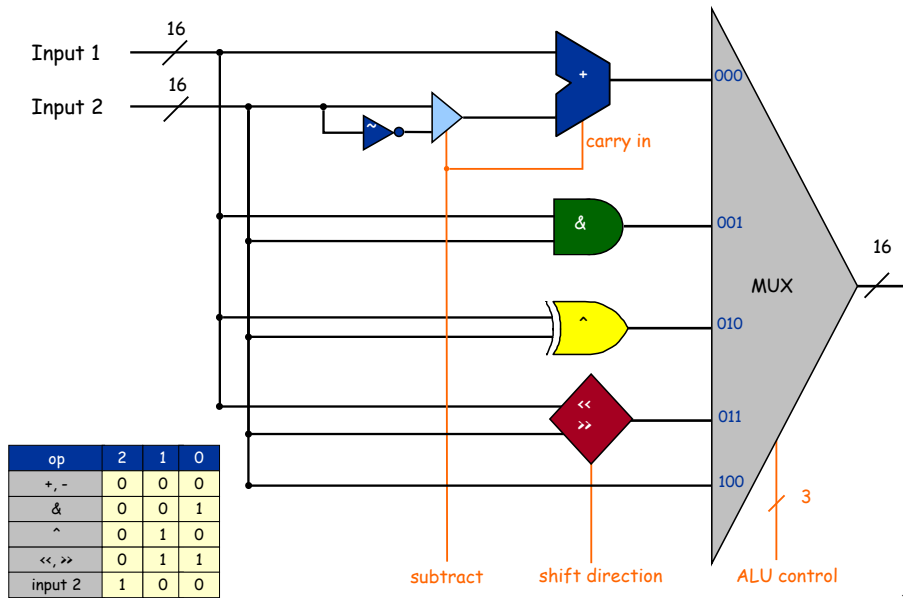
27

Stand-Alone Registers



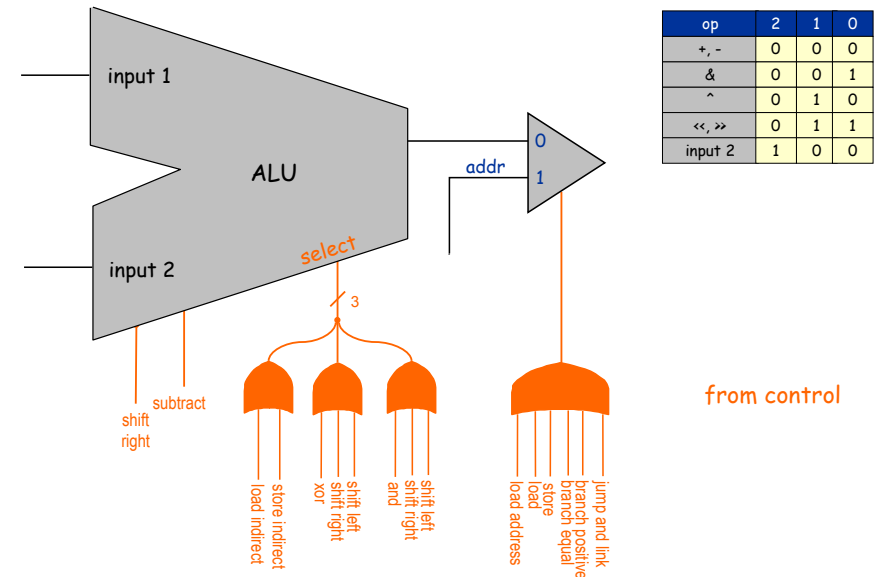
28

Arithmetic Logic Unit: Implementation



29

ALU Control



30

Layers of Abstraction

Abstraction	Built From	Examples
Abstract Switch	raw materials	transistor
Connector	raw materials	wire
Clock	raw materials	crystal oscillator
Logic Gates	abstract switches, connectors	AND, OR, NOT
Combinational Circuit	logic gates, connectors	decoder, multiplexer, adder, ALU
Sequential Circuit	logic gates, clock, connector	flip-flop
Components	decoder, multiplexer, adder, flip-flop	registers, ALU, counter, control
Computer	components	TOY

31

History + Future

Computer constructed by layering abstractions.

- Better implementation at low levels improves EVERYTHING.
- Ongoing search for better abstract switch!

History.

- 1820s: mechanical switches (Babbage's difference engine).
- 1940s: relays, vacuum tubes.
- 1950s: transistor, core memory.
- 1960s: integrated circuit.
- 1970s: microprocessor.
- 1980s: VLSI.
- 1990s: integrated systems.
- 2000s: web computer.
- Future: DNA, quantum, optical soliton, ...

32