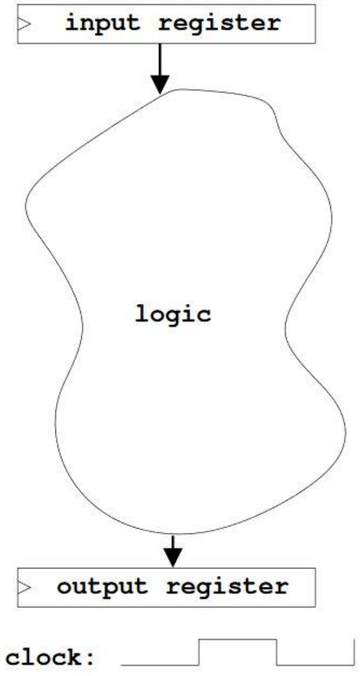
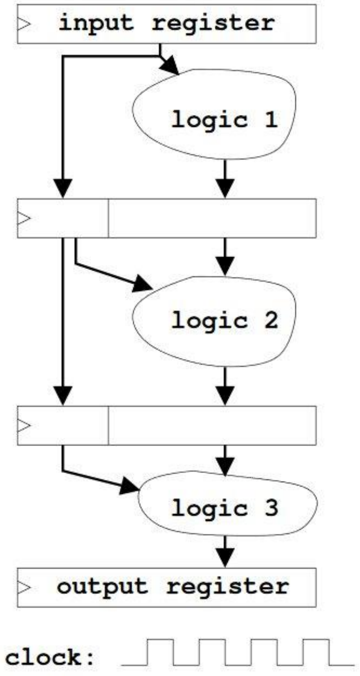
Tutorial 11

December 4, 2018

Speed up the β-machine with pipelines

The concept of pipeline is useful to increase the rate at which a deep combinatorial circuit can process the data





$$MIPS = \frac{Freq}{CPI}$$

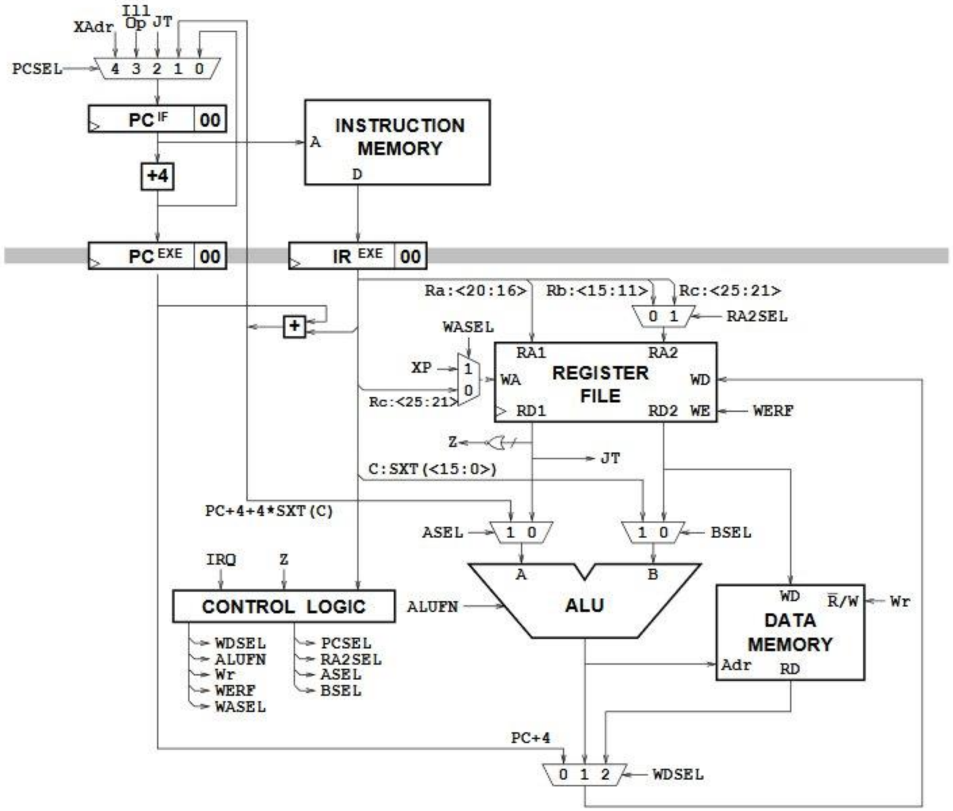
- **Terminology:**
  - **MIPS**: millions of instruction per second
  - **Freq**: clock frequency
  - **CPI**: clock (cycle) per instruction
- To increase performances we need either to **increase the frequency** or to **decrease the CPI**
- We cannot increase the frequency because we are blocked by the stabilization time of the circuit
- We can decrease the CPI $\Rightarrow$ pipelining

Pipelining the β-machine

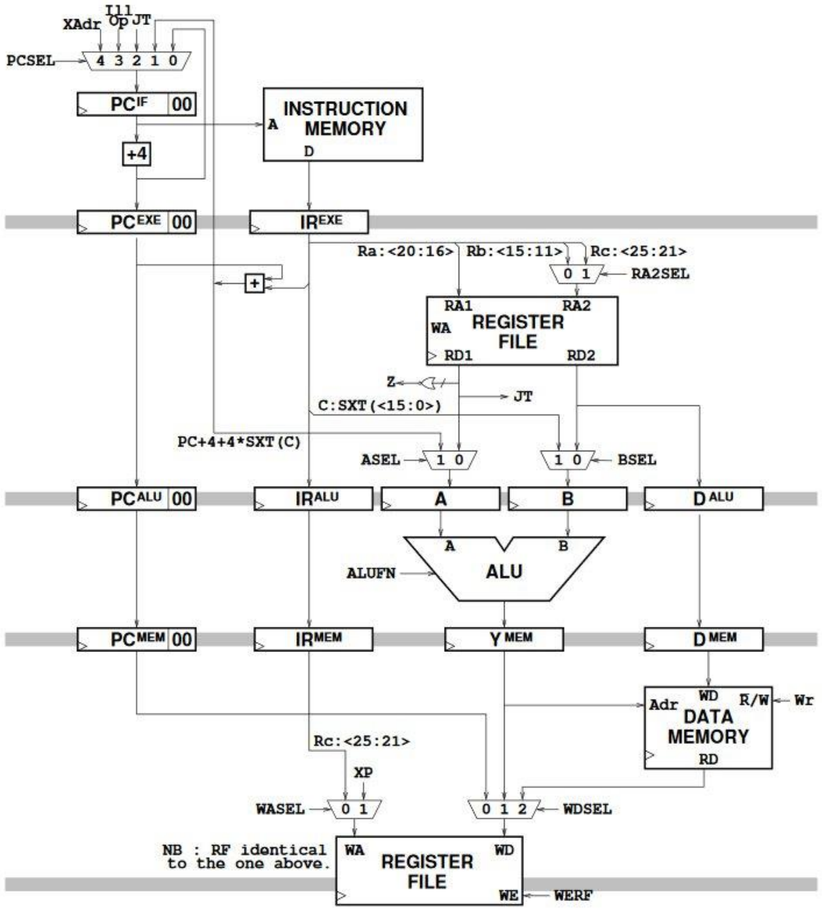
- We can decompose the combinatorial logic into five identifiable stages:
  - **Instruction fetch (IF)**: fetch the instruction from memory
  - **Register fetch (RF)**: fetch the arguments from the registers
  - **ALU**: computation of the ALU
  - **Memory (MEM)**: reading data from memory
  - **Write back (WB)**: writing back the results to the registers
- During this tutorial, we will study simplified versions with two stages and four stages (see next slide)

Pipelining the β-machine

A β machine with a 2-stage pipeline



2 stages: {IF}, {RF, ALU, MEM, WB}



4 stages: {IF}, {RF}, {ALU}, {MEM, WB}

Pipelining comes at a cost…

- With pipelines, we are able to speed the machine…
- … but as we process several instruction at once, some new problems occur
- During this tutorial, we are going to review some of those problems and evaluate their possible solutions

Two-stages pipeline

1. Give a plausible implementation of NOP().
2. Give a *software* solution to the jump problems for each of the following programs:

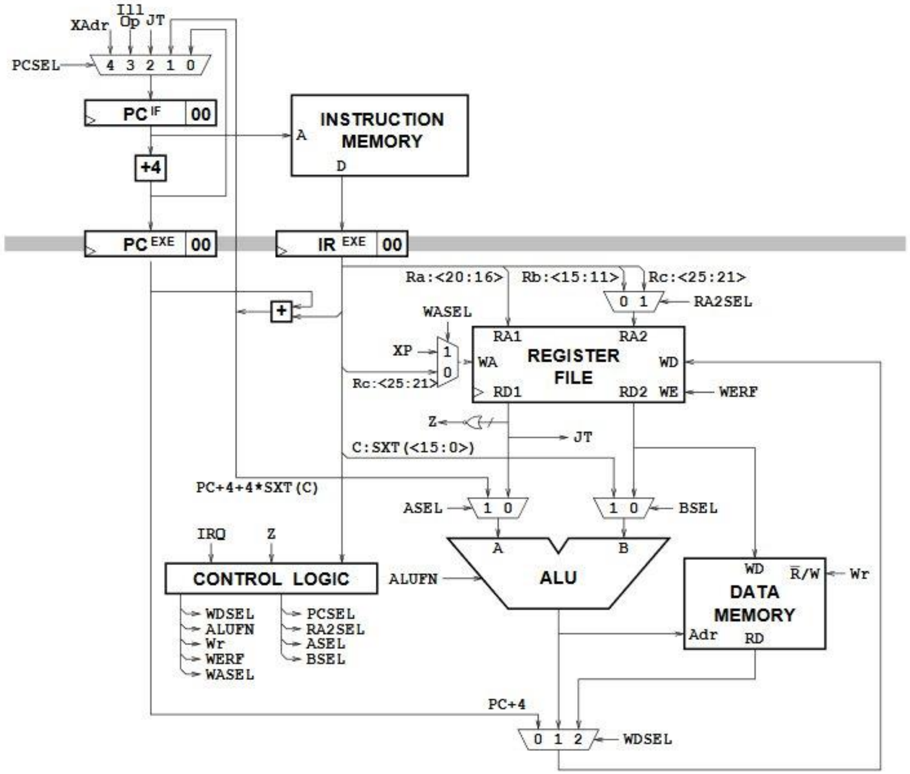
(a)

```
Main: ADDC(R31,0,R1)
      ADDC(R31,2,R2)
Incr: ADDC(R1,5,R1)
      SUBC(R2,1,R2)
      BT(R2,Incr)
Oper: ADD(R1,R3,R3)
```

(b)

```
Cas1: ADDC(R1,4,R1)
      SUBC(R2,12,R2)
      CMPLT(R1,R2,R0)
      BNE(R0,Cas2)
      MULC(R1,5,R1)
      BR(rtn)
Cas2: MULC(R2,5,R2)
      BR(rtn)
```

A β machine with a 2-stage pipeline



Four-stages pipeline

1. Give a software *and* a hardware solution to the data conflicts problems for the following program:

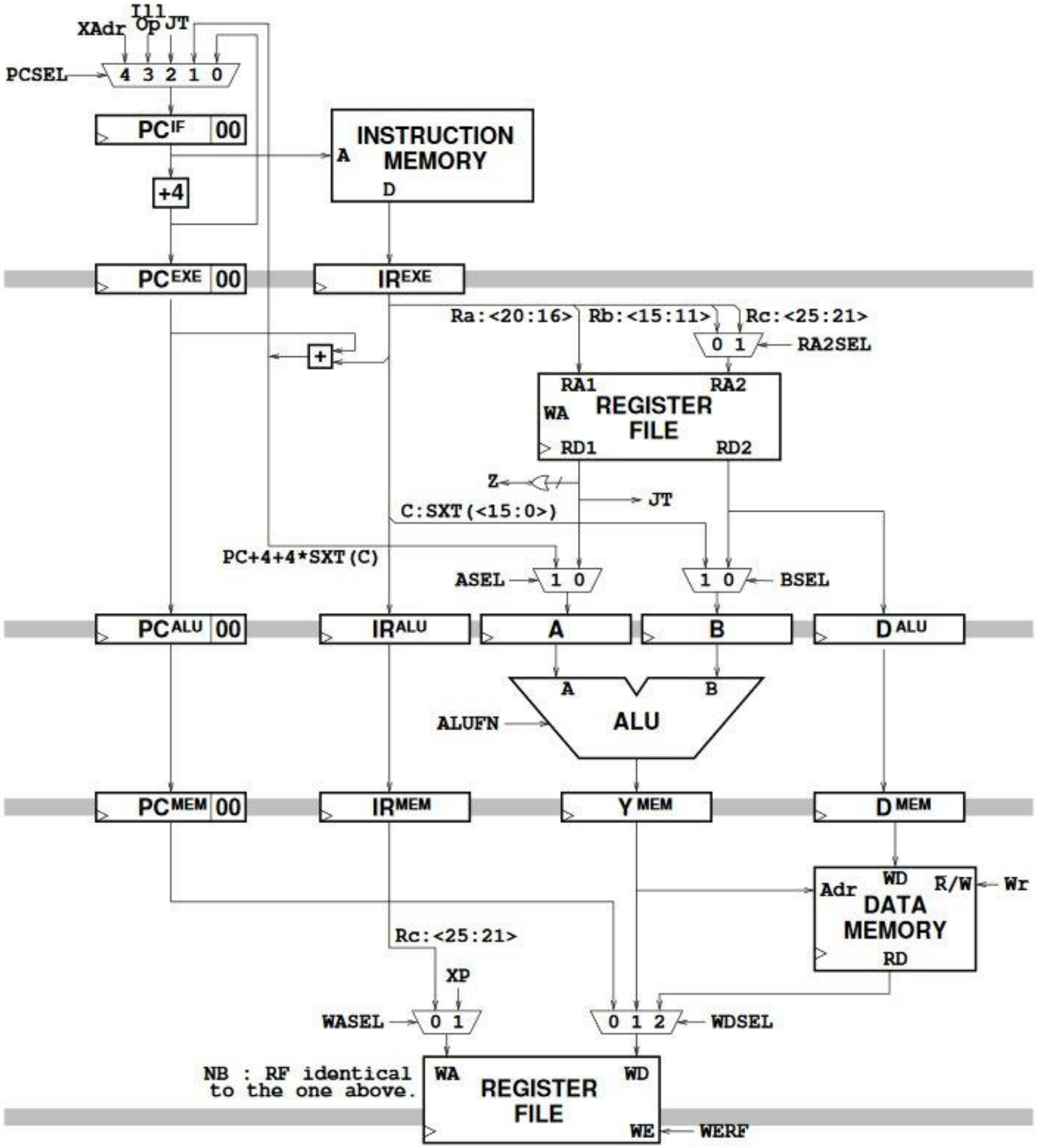
```
ADD(R1,R2,R3)
SUB(R3,R4,R5)
MULC(R2,5,R17)
ADD(R5,R1,R1)
SUB(R17,R1,R17)
```

2. Give a *hardware* solution to the data conflicts problems for the following program:

```
LD(R1,0,R4)
ADD(R1,R4,R5)
XOR(R3,R4,R6)
```

3. If the β Machine features 2 *bypasses*, what will be the result stored at 0x1000 after the execution of the following program? Why?

```
ADDC(R31,3,R0)
SUBC(R0,1,R1)
MUL(R0,R1,R2)
XOR(R0,R2,R3)
ST(R3,0x1000,R31)
```



Hardware solutions