CS 343

Simplified Single Cycle CPU with ADD/SUB instructions only

in VHDL and Model-SIM , Quartus, DE 2-70

March 27, 2015

Instructor: Professor Izidor Gertner

Design components:

1. 16 bit Instruction Register to store 16 bit instruction
2. 16 bit Program Counter Register
3. 16 16 bit word Register File
4. ADD/Sub instruction DATAPATH
5. Instruction decode unit to decode 2 bit opcode to required control signals ALUctr RegWr

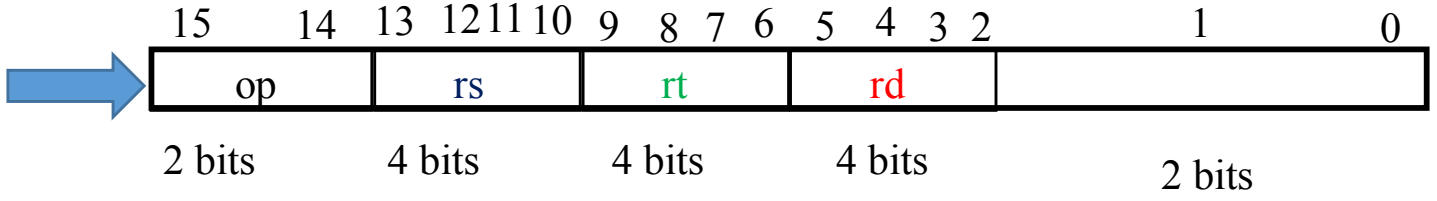
Optional : Instruction FETCH UNIT, BEQ instruction

Lab Assignments

Use the same techniques you have used to build SRAM in the previous lab.

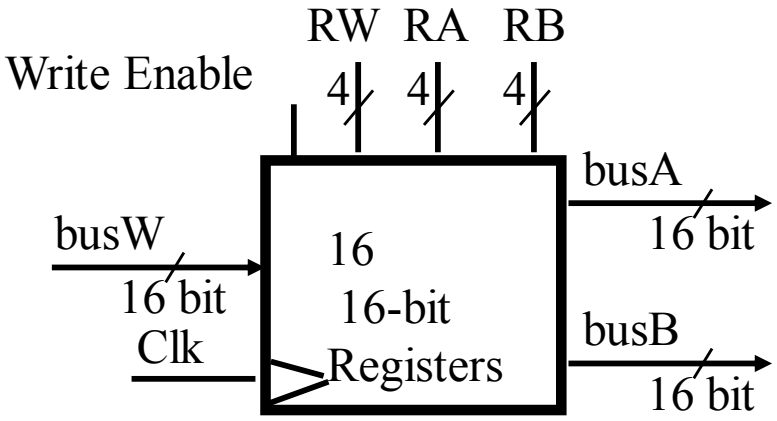
Design 16 bit Instruction Register, to store the currently executing instruction.

Instruction Register
Stores the instruction
During the execution
time



EXTEND THE SRAM DESIGN YOU HAVE SUBMITTED on MARCH 20 to REGISTER FILE

- Register File consists of 16 registers:
  - Two 16-bit output busses: busA and busB
  - One 16 -bit input bus: busW



- Register is selected by:
  - RA (4 bit number) selects the register to put on busA (data)
  - RB (4 bit number) selects the register to put on busB (data)
  - RW (4 bit number) selects the register to be written via busW (16 bit data) when Write Enable is 1
- Clock input (CLK)
  - The CLK input is a factor ONLY during write operation
  - ***During read operation, Register File behaves as a combinational logic block:***
    - RA or RB valid => busA or busB valid after "access time."

Add & Subtract Instruction

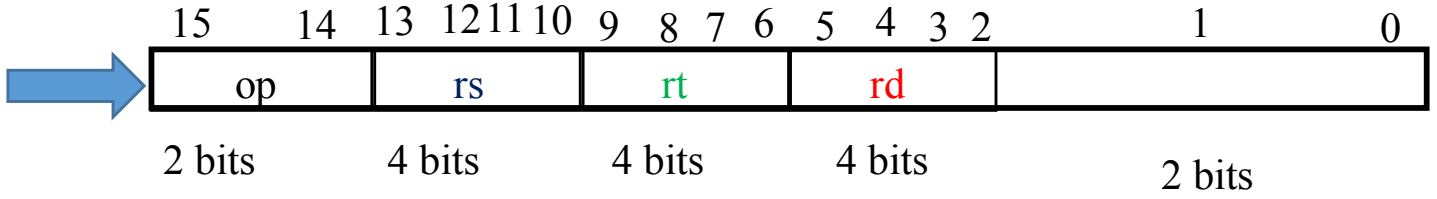
Add operation opcode bit[15]=0, bit[14]=0

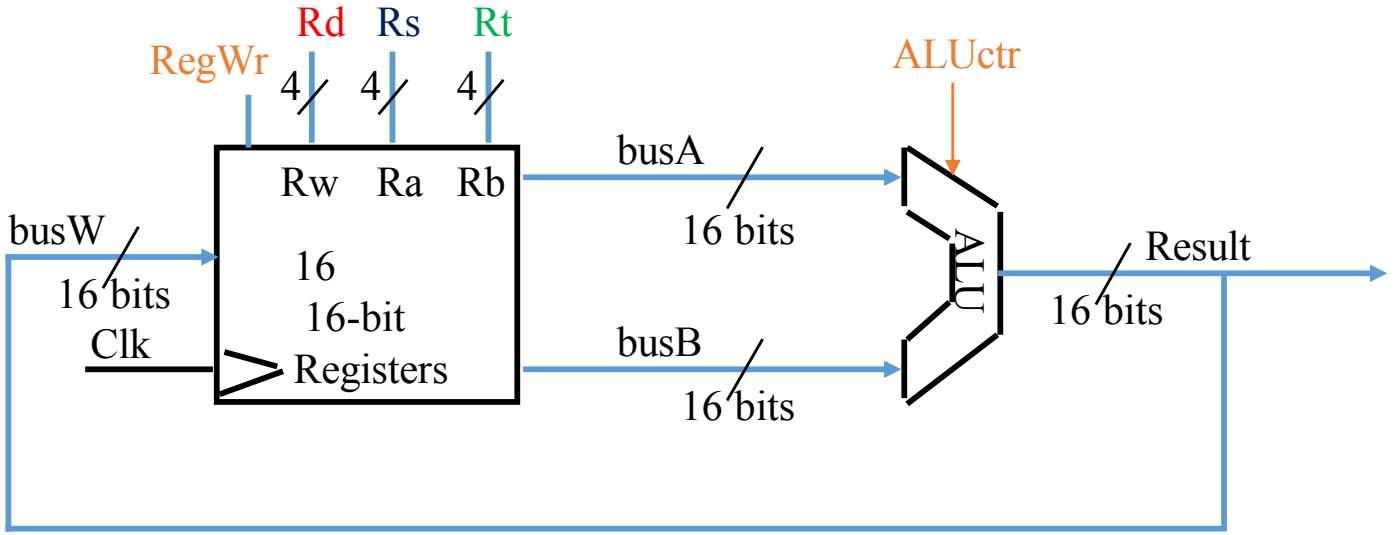
Sub operation opcode bit[15]=0, bit[14]=1

Control signal ALUctr =1 for SUB operation
Control signal ALUctr =0 for ADD operation

Control signal RegWrr =1 for write register **rd** operation

Instruction Register
Stores the instruction
During the execution time



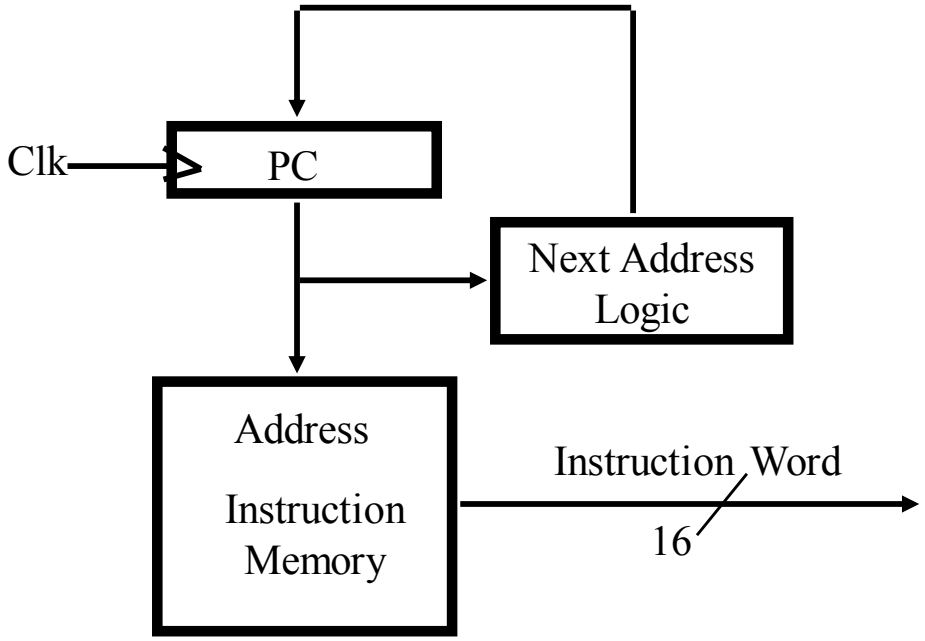


PC Register

Design PC Register.
Stores the instruction address
During the execution time

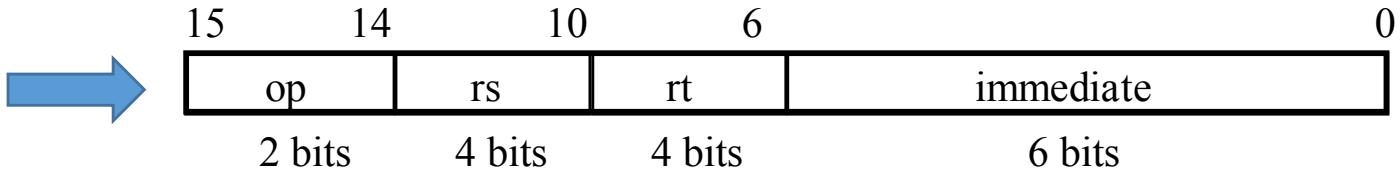
Optional Instruction Fetch Unit

- The common operations
  - Fetch the Instruction: mem[PC]
  - Update the program counter:
    - Sequential Code: PC <= PC + 4
    - Branch and Jump: PC <= “something else”



The Branch Instruction (optional)

Instruction Register
Stores the instruction
During the execution
time



- beq rs, rt, imm6
  - mem[PC] Fetch the instruction from memory
  - Equal <= (R[rs] == R[rt]) Calculate the branch condition
  - if (Equal) Calculate the next instruction's address
    - PC <= PC + 4 + { SignExt(imm6) , 2b00 }
  - else
    - PC <= PC + 4

Datapath for Branch Operations (optional)

- beq rs, rt, imm16

Datapath generates condition (equal)

Instruction Register
Stores the instruction
During the execution
time