**Control Unit**

CPU hardware that controls instruction execution

- sends signals to the datapath to operate it
- specifies what operations to perform, what data to move, when to move it, where to move it
Control Signals

Many control signals driven by the instruction

\[
\text{[ opcode ] [rs][rt][rd][shamt][func] \hspace{1em} R-type}
\]

\[
\text{[ opcode ] [rs][rt][ displacement ] \hspace{1em} I-type}
\]

\[
\text{[ opcode ] [ address ] \hspace{1em} J-type}
\]

Regularity of the MIPS formats

- **opcode** always in bits 31-26 (Op[5-0])
- source registers are always rs & rt
- base register always rs
- branch offset always bits 15-0
Our R2000 Control Signals

Register file
- register write signal: **RegWrite** asserted for R-type instructions & load
- register destination field: **RegDst** rt, rd
- results value: MemToReg loaded value, R-type instruction result
- all generated by the opcode

ALU
- type of the second operand: **ALUSrc** register, immediate
- generated by the opcode
- ALU operation: **ALUOp** add, subtract, and, or, set-on-less-than
  - generated by a small control unit
  - inputs: opcode & func field
  - output: ALU operation
  - examples:
    lw/sw => add
    beq => subtract
    R-type instruction => func value
Our R2000 Control Signals

Memory
- read signal: MemRead
- write signal: MemWrite
  - both generated by the opcode

Branch control
- new PC value: PCSrc
  incremented PC, target address
- generated by the opcode AND’d with Zero

Jump control
- new PC value:
  incremented PC or target address, jump address
  - generated by the opcode
Changing the Implementation

How should you approach a problem in which you had to redesign the implementation to include another instruction?

- What does the instruction do?
- What parts of the datapath does it need?
  - Can it use what is there already?
  - What new logic or registers does it need?
- How is the datapath activated?
  - What control lines does it need
  - Where should the control lines come from?