Déjà vu all over again!

As of 2am 2004-11-03, it looks like Bush was ahead but hadn’t yet clinched it. We may have to wait for a recount and an Ohio tabulation of provisional ballots. A Country Divided!
Review: How to Design a Processor: step-by-step

• 1. Analyze instruction set architecture (ISA) => datapath requirements
  • meaning of each instruction is given by the register transfers
  • datapath must include storage element for ISA registers
  • datapath must support each register transfer

• 2. Select set of datapath components and establish clocking methodology

• 3. Assemble datapath meeting requirements

• 4. Analyze implementation of each instruction to determine setting of control points that effects the register transfer.

• 5. Assemble the control logic
Why do we have two dirty bits?

if (wEnb)
    if (writeReg!=4'h0)
        begin
            array[writeReg] = writeD;
            dirty1=1'b1;
            dirty2=1'b1;
        end
always @ (readReg1 or dirty1)
    begin
        readD1 = array[readReg1];
        dirty1=0;
    end
always @ (readReg2 or dirty2)
    begin
        readD2 = array[readReg2];
        dirty2=0;
    end
Review: 3e: Store Operations

- Mem[R[rs] + SignExt[imm16]] = R[rt]

Ex.: `sw rt, rs, imm16`

```
<table>
<thead>
<tr>
<th>op</th>
<th>rs</th>
<th>rt</th>
<th>immediate</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 bits</td>
<td>5 bits</td>
<td>5 bits</td>
<td>16 bits</td>
</tr>
</tbody>
</table>
```

Diagram:

[Diagram showing the components of a CPU control unit with labels for registers, memory, and control signals.]
3f: The Branch Instruction

- `beq rs, rt, imm16`
  - `mem[PC]` Fetch the instruction from memory
  - `Equal = R[rs] == R[rt]` Calculate branch condition
  - `if (Equal) Calculate the next instruction’s address`
    - `PC = PC + 4 + (SignExt(imm16) x 4)`
  - `else`
    - `PC = PC + 4`
Datapath for Branch Operations

- \texttt{beq rs, rt, imm16}

Datapath generates condition (equal)

\begin{align*}
\text{op} & \quad \text{rs} & \quad \text{rt} & \quad \text{immediate} \\
6 \text{ bits} & \quad 5 \text{ bits} & \quad 5 \text{ bits} & \quad 16 \text{ bits}
\end{align*}

- Already MUX, adder, sign extend, zero
Putting it All Together: A Single Cycle Datapath

Instruction<31:0> → Inst Memory

Inst Memory

Rs → ALUctr
Rt → ALUctr
Rd → ALUctr
Imm16 → ALUctr

nPC sel → Adder

Adder

RegDst

[Rd | Rt] → Mux

Mux

Equal → ALUctr

ALUctr

MemWr

MemtoReg

Rw Ra Rb

32 32-bit Registers

busA

busB

32

32

Data In

WrEnAdr

Data Memory

ExtOp

ALUSrc

Clk

Imm16

16

Extender

Mux

PC Ext

imm16

4

Adder

Adder

Adder

Mux

Clk

1

PC

Clk

4

nPC sel

RegWr

5

5

5

Rs

Rt

Clk
An Abstract View of the Critical Path

Critical Path (Load Operation) =
- Delay clock through PC (FFs) +
- Instruction Memory’s Access Time +
- Register File’s Access Time +
- ALU to Perform a 32-bit Add +
- Data Memory Access Time +
- Stable Time for Register File Write

- This affects how much you can overclock your PC!

32 32-bit Registers

Data Memory

Instruction Address

Next Address

PC

Ideal Instruction Memory

Instruction

Rd, Rs, Rt

Imm

A, B

32

32

32

32

Clk

Ideal Data Memory

Data Address

Data In

Clk
An Abstract View of the Implementation

Datapath

Control

Conditions

Control Signals

Ideal
Instruction
Memory

Instruction Address

Next Address

PC

Clk

Instruction

Rd
d5

Rs
d5

Rt
d5

Rw
d32
Ra
d32
Rb
d32

32 32-bit Registers

A

B

32

32

Data Address

Data In

ALU

Data Out

Ideal
Data
Memory

Cal

CS 61C L28 Single Cycle CPU Control I (9)
Summary: A Single Cycle Datapath

- Rs, Rt, Rd, Imed16 connected to datapath
- We have everything except control signals
Anatomy Review: 5 components of any Computer

- **Personal Computer**
- **Processor** (Control (“brain”))
- **Datapath** (“brawn”)
- **Memory** (where programs, data live when running)
- **Devices**
  - **Input**
  - **Output**

Yesterday (& finish up)

Today

Keyboard, Mouse

Disk (where programs, data live when not running)

Display, Printer
Recap: Meaning of the Control Signals

- **nPC_MUX_sel:**
  - $0 \Rightarrow PC \leftarrow PC + 4$
  - $1 \Rightarrow PC \leftarrow PC + 4 + \{\text{SignExt(Im16)}, 00\}$
  - "n" = next

- Later in lecture: higher-level connection between mux and branch condition

![Diagram of control signals](image-url)
Recap: Meaning of the Control Signals

- **ExtOp**: “zero”, “sign”
  - MemWr: 1 ⇒ write memory
- **ALUsrc**: 0 ⇒ regB; 1 ⇒ immed
  - MemtoReg: 0 ⇒ ALU; 1 ⇒ Mem
  - RegDst: 0 ⇒ “rt”; 1 ⇒ “rd”
- **ALUctr**: “add”, “sub”, “or”
  - RegWr: 1 ⇒ write register
Great talk today – Don’t miss

306 Soda Hall @ 4pm

• Dr. David Anderson
  • Space Sciences Laboratory, U.C. Berkeley. SETI Director

“Public Resource Computing”

The majority of the world's computing power is no longer concentrated in supercomputer centers and machine rooms. Instead it is distributed around the world in hundreds of millions of personal computers and game consoles, many connected to the Internet. A new computing paradigm, “public-resource computing”, uses these PCs to do scientific supercomputing. This paradigm enables new research in a number of areas and has social implications as well: it catalyzes global communities centered around common interests and goals, it encourages public awareness of current scientific research, and it may give the public a measure of control over the directions of science progress.
Administrivia

- Dan will be away at a conference on Thursday and Friday, Andrew will cover lecture.

- We regraded all the midterms and your TAs have them to return to you.
RTL: The Add Instruction

```
<table>
<thead>
<tr>
<th>31</th>
<th>26</th>
<th>21</th>
<th>16</th>
<th>11</th>
<th>6</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>op</td>
<td>rs</td>
<td>rt</td>
<td>rd</td>
<td>shamt</td>
<td>funct</td>
<td></td>
</tr>
<tr>
<td>6 bits</td>
<td>5 bits</td>
<td>5 bits</td>
<td>5 bits</td>
<td>5 bits</td>
<td>6 bits</td>
<td></td>
</tr>
</tbody>
</table>
```

**add rd, rs, rt**

- **MEM[PC]**
  - Fetch the instruction from memory

- **R[rd] = R[rs] + R[rt]**
  - The actual operation

- **PC = PC + 4**
  - Calculate the next instruction’s address
Instruction Fetch Unit at the Beginning of Add

- Fetch the instruction from Instruction memory: Instruction = MEM[PC]
  - same for all instructions

```
Instruction Fetch Unit

Inst Memory Addr

Instruction<31:0>

nPC_MUX_sel

imm16

4

Adder

Adder

Adder

Mux

PC Ext

PC

Clk

[Diagram]
```
The Single Cycle Datapath during **Add**

- **R[rd] = R[rs] + R[rt]**

### Instruction Fetch Unit
- Instruction<31:0>
- **nPC_sel= +4**

### ALU Unit
- ALUctr = Add
- **ALUSrc = 0**
- **ExtOp = x**

### Memory Unit
- MemWr = 0
- MemtoReg = 0
- **Zero**
- **WrEn**
- **Adr**
- **Data Memory**
- **Data In**

### Register File
- **32 32-bit Registers**
- **RegDst = 1**
- **RegWr = 1**
- **busA**
- **busB**
- **imm16**

### Control Signals
- **Clk**
- **busW**
- **Rw**
- **Ra**
- **Rb**
- **Rs**
- **Rt**

### Other Signals
- **Extender**
- **Zero**
- **Zero**
- **Imm16**
Instruction Fetch Unit at the End of Add

- \( \text{PC} = \text{PC} + 4 \)
- This is the same for all instructions except: Branch and Jump
Suppose we’re writing a MIPS interpreter in Verilog. Which sequence below is best organization for the interpreter?

A. repeat loop that fetches instructions
B. while loop that fetches instructions
C. Decodes instructions using case statement
D. Decodes instr. using chained if statements
E. Executes each instruction
F. Increments PC by 4

1: ACEF
2: ADEF
3: AECF
4: AEDF
5: BCEF
6: BDEF
7: BECF
8: BEDF
9: EF
0: FAE
Summary: Single cycle datapath

° 5 steps to design a processor
  • 1. Analyze instruction set => datapath requirements
  • 2. Select set of datapath components & establish clock methodology
  • 3. Assemble datapath meeting the requirements
  • 4. Analyze implementation of each instruction to determine setting of control points that effects the register transfer.
  • 5. Assemble the control logic

° Control is the hard part

° MIPS makes that easier
  • Instructions same size
  • Source registers always in same place
  • Immediates same size, location
  • Operations always on registers/immediates