CPU design involves Datapath, Control
  1. Instruction Fetch
  2. Instruction Decode & Register Read
  3. ALU (Execute)
  4. Memory
  5. Register Write

Datapath timing: single long clock cycle or one short clock cycle per stage
Datapath and Control

- Datapath based on data transfers required to perform instructions
- Controller causes the right transfers to happen
Five Components of a Computer

Computer

- Processor
  - Control
- Memory (passive)
  - (where programs, data live when running)
- Datapath
- Devices
  - Input
  - Output
- Keyboard, Mouse
- Disk (where programs, data live when not running)
- Display, Printer
Processor Design: 5 Steps

1: Analyze instruction set to determine datapath requirements
   ▪ Meaning of each instruction is given by register transfers
   ▪ Datapath must include storage element for ISA registers
   ▪ Datapath must support each register transfer

2: Select set of datapath components & establish clock methodology

3: Assemble datapath components that meet the requirements

4: Analyze implementation of each instruction to determine setting of control points that realizes the register transfer

5: Assemble the control logic
The MIPS Instruction Formats

- All MIPS instructions are 32 bits long. 3 formats:

<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>
<tr>
<td>31</td>
<td>26</td>
<td>21</td>
<td>16</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>26</th>
<th>21</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>op</td>
<td>rs</td>
<td>rt</td>
<td>address / immediate</td>
</tr>
<tr>
<td>6 bits</td>
<td>5 bits</td>
<td>5 bits</td>
<td>16 bits</td>
</tr>
<tr>
<td>31</td>
<td>26</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>op</td>
<td>target address</td>
</tr>
<tr>
<td>6 bits</td>
<td>26 bits</td>
</tr>
</tbody>
</table>

- The different fields are:
  - op: operation ("opcode") of the instruction
  - rs, rt, rd: the source and destination register specifiers
  - shamt: shift amount
  - funct: selects the variant of the operation in the "op" field
  - address / immediate: address offset or immediate value
  - target address: target address of jump instruction
The MIPS-lite Subset

- **ADDU and SUBU**
  - `addu rd, rs, rt`
  - `subu rd, rs, rt`

- **OR Immediate**
  - `ori rt, rs, imm16`

- **LOAD and STORE Word**
  - `lw rt, rs, imm16`
  - `sw rt, rs, imm16`

- **BRANCH**
  - `beq rs, rt, imm16`

\[
\begin{array}{cccccc}
31 & 26 & 21 & 16 & 11 & 6 \\
\hline
\text{op} & \text{rs} & \text{rt} & \text{rd} & \text{shamt} & \text{funct} \\
6 \text{ bits} & 5 \text{ bits} & 5 \text{ bits} & 5 \text{ bits} & 5 \text{ bits} & 6 \text{ bits} \\
\end{array}
\]

\[
\begin{array}{cccc}
31 & 26 & 21 & 16 \\
\hline
\text{op} & \text{rs} & \text{rt} & \text{address / immediate} \\
6 \text{ bits} & 5 \text{ bits} & 5 \text{ bits} & 16 \text{ bits} \\
\end{array}
\]

\[
\begin{array}{cccc}
31 & 26 & 21 & 16 \\
\hline
\text{op} & \text{rs} & \text{rt} & \text{address / immediate} \\
6 \text{ bits} & 5 \text{ bits} & 5 \text{ bits} & 16 \text{ bits} \\
\end{array}
\]

\[
\begin{array}{cccc}
31 & 26 & 21 & 16 \\
\hline
\text{op} & \text{rs} & \text{rt} & \text{address / immediate} \\
6 \text{ bits} & 5 \text{ bits} & 5 \text{ bits} & 16 \text{ bits} \\
\end{array}
\]
Register Transfer Language (RTL)

- RTL gives the **meaning** of the instructions
- All start by fetching the instruction

\[
\begin{align*}
\{ \text{op} , \ rs , \ rt , \ rd , \ \text{shamt} , \ \text{funct} \} & \leftarrow \ \text{MEM}[ \ PC ] \\
\{ \text{op} , \ rs , \ rt , \ \text{Imm16} \} & \leftarrow \ \text{MEM}[ \ PC ] \\
\# \ \text{Inst} & \ \text{Register Transfers} \\
\text{ADDU} & : \ R[rd] \leftarrow R[rs] + R[rt]; \ PC \leftarrow PC + 4 \\
\text{SUBU} & : \ R[rd] \leftarrow R[rs] - R[rt]; \ PC \leftarrow PC + 4 \\
\text{ORI} & : \ R[rt] \leftarrow R[rs] \mid \text{zero_ext}(\text{Imm16}); \ PC \leftarrow PC + 4 \\
\text{LOAD} & : \ R[rt] \leftarrow \ \text{MEM}[ \ R[rs] + \text{sign_ext}(\text{Imm16})]; \ PC \leftarrow PC + 4 \\
\text{STORE} & : \ \text{MEM}[ \ R[rs] + \text{sign_ext}(\text{Imm16}) ] \leftarrow R[rt]; \ PC \leftarrow PC + 4 \\
\text{BEQ} & : \ \text{if ( } R[rs] == R[rt] \ \text{) } \\
\text{then} \ PC & \leftarrow PC + 4 + (\text{sign_ext}(\text{Imm16}) || 00) \\
\text{else} \ PC & \leftarrow PC + 4
\end{align*}
\]
Step 1: Requirements of the Instruction Set

- Memory (MEM)
  - Instructions & data (will use one for each)
- Registers (R: 32 × 32)
  - Read RS
  - Read RT
  - Write RT or RD
- PC
- Extender (sign/zero extend)
- Add/Sub/OR unit for operation on register(s) or extended immediate
- Add 4 (+ maybe extended immediate) to PC
- Compare registers?
Step 2: Components of the Datapath

- Combinational Elements
- Storage Elements + Clocking Methodology
- Building Blocks
ALU Needs for MIPS-lite + Rest of MIPS

- Addition, subtraction, logical OR, \(==\):
  
  ADDU \( R[rd] = R[rs] + R[rt]; \ldots \)
  
  SUBU \( R[rd] = R[rs] - R[rt]; \ldots \)
  
  ORI \( R[rt] = R[rs] \mid \text{zero\_ext(Imm16)}\ldots \)
  
  BEQ \( \text{if } ( R[rs] == R[rt] )\ldots \)

- Test to see if output \(==0\) for any ALU operation gives \(==\) test. How?

- P&H also adds AND, Set Less Than (1 if \(A < B\), 0 otherwise)

- ALU follows Chapter 5
Storage Element: Idealized Memory

- Memory (idealized)
  - One input bus: Data In
  - One output bus: Data Out

- Memory word is found by:
  - Address selects the word to put on Data Out
  - Write Enable = 1: address selects the memory word to be written via the Data In bus

- Clock input (CLK)
  - CLK input is a factor ONLY during write operation
  - During read operation, behaves as a combinational logic block: Address valid ⇒ Data Out valid after “access time”
Storage Element: Register (Building Block)

- Similar to D Flip Flop except
  - N-bit input and output
  - Write Enable input

- Write Enable:
  - Negated (or deasserted) (0): Data Out will not change
  - Asserted (1): Data Out will become Data In on positive edge of clock
Register File consists of 32 registers:
- Two 32-bit output busses: busA and busB
- One 32-bit input bus: busW

Register is selected by:
- RA (number) selects the register to put on busA (data)
- RB (number) selects the register to put on busB (data)
- RW (number) selects the register to be written via busW (data) when Write Enable is 1

Clock input (clk)
- Clk input is a factor ONLY during write operation
- During read operation, behaves as a combinational logic block: RA or RB valid $\Rightarrow$ busA or busB valid after “access time”.
Step 3a: Instruction Fetch Unit

- Register Transfer Requirements ⇒ Datapath Assembly
- Instruction Fetch
- Read Operands and Execute Operation
- Common RTL operations
  - Fetch the Instruction: mem[PC]
  - Update the program counter:
    - Sequential Code: PC ← PC + 4
    - Branch and Jump: PC ← “something else”

Ning Xie (http://www.csd.uwo.ca/courses/CS3350b) Department of Computer Science University of Western Ontario, Canada
Step 3b: Add & Subtract

\[ R[rd] = R[rs] \text{ op } R[rt] \] (addu rd, rs, rt)

- Ra, Rb, and Rw come from instruction’s Rs, Rt, and Rd fields
  
  \[
  \begin{array}{cccccc}
  \text{op} & \text{rs} & \text{rt} & \text{rd} & \text{shamt} & \text{funct} \\
  \text{6 bits} & \text{5 bits} & \text{5 bits} & \text{5 bits} & \text{5 bits} & \text{6 bits}
  \end{array}
  \]

- ALUctr and RegWr: control logic after decoding the instruction

- The register file & ALU are already defined
Clocking Methodology

- Storage elements clocked by same edge
- Flip-flops (FFs) and combinational logic have some delays
  - Gates: delay from input change to output change
  - Signals at FF D input must be stable before active clock edge to allow signal to travel within the FF (set-up time), and we have the usual clock-to-Q delay
- “Critical path” (longest path through logic) determines length of clock period
Register-Register Timing: One Complete Cycle

- Clk
- PC: Old Value → New Value
- Rs, Rt, Rd: Instruction Memory Access Time
- Op, Func: Old Value → New Value
- ALUctr: Old Value → New Value
- RegWr: Old Value → New Value
- busA, B: Register File Access Time
- busW: ALU Delay
- RegFile
- ALUctr
- Register Write Occurs Here
Putting it All Together: A Single Cycle Datapath

Ning Xie (http://www.csd.uwo.ca/courses/CS3350B) Department of Computer Science University of Western Ontario, Canada

Concluding Remarks: 3 of 5 Steps of Processor Design

1: Analyze instruction set to determine datapath requirements
   • Meaning of each instruction is given by register transfers
   • Datapath must include storage element for ISA registers
   • Datapath must support each register transfer

2: Select set of datapath components & establish clock methodology

3: Assemble datapath components that meet the requirements

4: Analyze implementation of each instruction to determine setting of control points that realizes the register transfer

5: Assemble the control logic