Processor Design — Single Cycle Processor

Hung-Wei Tseng
Recap: the stored-program computer

- Store instructions in memory
- The program counter (PC) controls the execution
Recap: MIPS ISA

- **R-type**: add, sub, and etc...
  - 6 bits: opcode
  - 5 bits: rs
  - 5 bits: rt
  - 5 bits: rd
  - 5 bits: shift amount
  - 6 bits: funct

- **I-type**: addi, lw, sw, beq, and etc...
  - 6 bits: opcode
  - 5 bits: rs
  - 5 bits: rt
  - 16 bits: immediate / offset

- **J-type**: j, jal, and etc...
  - 6 bits: opcode
  - 26 bits: target
Outline

• Implementing a MIPS processor
  • Single-cycle processor
  • Pipelined processor
Designing a simple MIPS processor

• Support MIPS ISA in hardware
  • Design the datapath: add and connect all the required elements in the right order
  • Design the control path: control each datapath element to function correctly.

• Starts from designing a single cycle processor
  • Each instruction takes exactly one cycle to execute
Basic steps of execution

- Instruction fetch: fetch an instruction from memory
- Decode:
  - What’s the instruction?
  - Where are the operands?
- Execute
- Memory access
  - Where is my data? (The data memory address)
- Write back
  - Where to put the result
- Determine the next PC

```
Processor

ALU

PC

R0
R1
R2
R3

 registers

120007a30: 0f00bb27  ldah gp,15(t12)
120007a34: 509cbd23  lda gp,-25520(gp)
120007a38: 00005d24  ldah t1,0(gp)
120007a3c: 0000bd24  ldah t4,0(gp)
120007a40: 2ca422a0  ldl t0,-23508(t1)
120007a44: 130020e4  beq t0,120007a94
120007a48: 00003d24  ldah t0,0(gp)
120007a4c: 2ca4e2b3  stl zero,-23508(t1)

800bf9000: 00c2e800  12773376
800bf9004: 00000008  8
800bf9008: 00c2f000  12775424
800bf900c: 00000008  8
800bf9010: 00c2f800  12777472
800bf9014: 00000008  8
800bf9018: 00c30000  12779520
800bf901c: 00000008  8
```
Recap: MIPS ISA

- **R-type**: add, sub, and etc...
  
<table>
<thead>
<tr>
<th>opcode</th>
<th>rs</th>
<th>rt</th>
<th>rd</th>
<th>shift amount</th>
<th>funct</th>
</tr>
</thead>
<tbody>
<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>
</tr>
</tbody>
</table>

- **I-type**: addi, lw, sw, beq, and etc...
  
<table>
<thead>
<tr>
<th>opcode</th>
<th>rs</th>
<th>rt</th>
<th>immediate / offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 bits</td>
<td>5 bits</td>
<td>5 bits</td>
<td>16 bits</td>
</tr>
</tbody>
</table>

- **J-type**: j, jal, and etc...
  
<table>
<thead>
<tr>
<th>opcode</th>
<th>target</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 bits</td>
<td>26 bits</td>
</tr>
</tbody>
</table>
Implementing an R-type instruction

- How many of the following datapath elements is necessary for an R-type instruction?
  I. Instruction Memory
  II. Data memory
  III. Register file
  IV. Program counter
  V. ALU

A. 1
B. 2
C. 3
D. 4
E. 5

instruction = MEM[PC]
REG[rd] = REG[rs] op REG[rt]
PC = PC + 4
Implementing an R-type instruction

• What’s right order of accessing the datapath elements for an R-type instruction?
  I. Instruction Memory
  II. Data memory
  III. Register file
  IV. Program counter
  V. ALU
  A. I, III, V, IV
  B. IV, I, III, V
  C. I, V, III, IV
  D. IV, V, I, III
  E. none of the above
Implementing an R-type instruction

instruction = MEM[PC]
REG[rd] = REG[rs] op REG[rt]
PC = PC + 4

Tell the ALU what ALU function to perform

Tell the Processor when to start an instruction
Implementing a load instruction

• How many of the following datapath elements is necessary for a load instruction?
  I. Instruction Memory
  II. Data memory
  III. Register file
  IV. Program counter
  V. ALU

A. 1
B. 2
C. 3
D. 4
E. 5

instruction = MEM[PC]
REG[rt] = MEM[signext(immediate) + REG[rs]]
PC = PC + 4

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

11
Implementing a load instruction

• What’s right order of accessing the datapath elements for a load instruction?
  I. Instruction Memory
  II. Data memory
  III. Register file
  IV. Program counter
  V. ALU
  
  A. IV, I, III, V, II
  B. IV, I, III, II, V
  C. IV, I, V, II, III
  D. IV, I, II, V, III
  E. none of the above
Implementing a load instruction

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

instruction = MEM[PC]
REG[rt] = MEM[signext(immediate) + REG[rs]]
PC = PC + 4

Set different control signals for different types of instructions
Set to 1 if it's a load
Set to 0 if it's a load
Implementing a store instruction

- How many of the following datapath elements is necessary for a store instruction?
  I. Instruction Memory
  II. Data memory
  III. Register file
  IV. Program counter
  V. ALU

A. 1
B. 2
C. 3
D. 4
E. 5

instruction = MEM[PC]
MEM[signext(immediate) + REG[rs]] = REG[rt]
PC = PC + 4
Implementing a store instruction

• What’s right order of accessing the datapath elements for a store instruction?
  I. Instruction Memory
  II. Data memory
  III. Register file
  IV. Program counter
  V. ALU

A. IV, I, III, V, II
B. IV, I, III, II, V
C. IV, I, V, II, III
D. IV, I, II, V, III
E. none of the above
Implementing a store instruction

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

instruction = MEM[PC]
MEM[signext(immediate) + REG[rs]] = REG[rt]
PC = PC + 4
Implementing a branch instruction

- How many of the following datapath elements is necessary for a branch instruction?
  I. Instruction Memory
  II. Data memory
  III. Register file
  IV. Program counter
  V. ALU

A. 1
B. 2
C. 3
D. 4
E. 5

- Instruction = MEM[PC]
- PC = (REG[rs] == REG[rt]) ? PC + 4 + SignExtImmediate * 4 : PC + 4
Implementing a branch instruction

(opcode, rs, rt, immediate / offset)

\[
\text{instruction} = \text{MEM}[\text{PC}]
\]

\[
\text{PC} = (\text{REG}[\text{rs}] == \text{REG}[\text{rt}]) \ ? \ \text{PC} + 4 + \text{SignExtImmediate} \times 4 : \text{PC} + 4
\]

PCSrc = Branch & Zero

Calculate the target address
Performance of a single-cycle processor

How many of the following statements about a single-cycle processor is correct?

- The CPI of a single-cycle processor is always 1
- If the single-cycle implements lw, sw, beq, and add instructions, the sw instruction determines the cycle time
- Hardware elements are mostly idle during a cycle
- We can always reduce the cycle time of a single-cycle processor by supporting fewer instructions

A. 0  
B. 1  
C. 2  
D. 3  
E. 4
Q & A