Control Related Instructions - Jumps

• jr rs ; jump register: PC ← Regs[rs]

• jalr rs,rd ; jump and link register:
  Regs[rd] ← [PC]+4; PC ← Regs[rs]

• j jump_target ; jump inside 256 MB region:
  PC low order 28 bits ← jump_target || 2 zero-bits

• jal jump_target ; jump inside 256 MB region and link:
  Regs[31] ← [PC]+4
  PC low order 28 bits ← jump_target || 2 zero-bits

Control Related Instructions - Branches

• beq rs, rt, offset ; branch on equal:
  if (Regs[rs] = Regs[rt])
  then PC ← [PC]+4+ 14-bit sign extend || offset || 2 zero-bits
  else PC ← [PC]+4

• bne rs, rt, offset ; branch on not equal:
  if (Regs[rs] != Regs[rt])
  then PC ← [PC]+4+ 14-bit sign extend || offset || 2 zero-bits
  else PC ← [PC]+4

• bgez rs, offset ; branch on greater or equal zero:
  if (Regs[rs] ≥ 0)
  then PC ← [PC]+4+ 14-bit sign-extend || offset || 2 zero-bits
  else PC ← [PC]+4

  Plus: bgtz, blez, bltz

Illustration of MIPS Addressing Modes

<table>
<thead>
<tr>
<th>Addressing Mode</th>
<th>Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>Immediate</td>
<td><img src="image" alt="Diagram of Immediate Addressing" /></td>
</tr>
<tr>
<td>Register</td>
<td><img src="image" alt="Diagram of Register Addressing" /></td>
</tr>
<tr>
<td>Base</td>
<td><img src="image" alt="Diagram of Base Addressing" /></td>
</tr>
<tr>
<td>PC-relative</td>
<td><img src="image" alt="Diagram of PC-relative Addressing" /></td>
</tr>
<tr>
<td>Pseudo-direct</td>
<td><img src="image" alt="Diagram of Pseudo-direct Addressing" /></td>
</tr>
</tbody>
</table>

Figure 2.24

Special Control - Related Instructions

• syscall ; to cause a syscall exception
  Encoding: 000000 00000000000000000000 001100

• break ; to cause a break exception
  Encoding: 000000 00000000000000000000 001101

• teq rs, rt ; trap exception if equal: if (Regs[rs] == Regs[rt])
  then trap exception

• tlti rs, immediate ; trap exception if less than immediate:
  if (Regs[rs] < 48-bit sign-extend || immediate
  then trap exception

• eret ; return from exception
  Plus: several additional conditional trap instructions
CPU Modes and Address Spaces

There are two processor (CPU) modes of operation:
- Kernel (Supervisor) Mode
- User Mode

The processor is in Kernel Mode when CPU mode bit in Status register is set to one. The processor enters Kernel Mode at power-up, or as result of an interrupt, exception, or error. The processor leaves Kernel Mode and enters User Mode when the CPU mode bit is set to zero (by some instruction).

Memory address space is divided in two ranges (simplified):
- User address space
  - addresses in the range \([0 – 7FFFFF_{16}]\)
- Kernel address space
  - addresses in the range \([80000000_{16} – FFFFFFFF_{16}]\)

Privilege Instructions

When operating in User Mode, processor has access only to the CPU and FPU registers, while when operating in Kernel Mode, processor has access to the full capabilities of processor including CP0 registers.

Privileged instructions can not be executed when the processor is in User mode, i.e. they can be executed only when the processor is in Kernel mode.

Examples of MIPS privileged instructions:
- any instruction that accesses Kernel address space,
- mfc0 – move word from CP0 to CPU,
- mtc0 – move word to CP0 from CPU,
- lw0 – load (from memory) word into CP0,
- swc0 – store (into memory) word from CP0.

MIPS Exceptions: A Subset

There are four sets of causes for an exception.

A. Exceptions caused by hardware malfunctioning:
- Machine Check: Processor detects internal inconsistency;
- Bus Error: on a load or store instruction, or instruction fetch;

B. Exceptions caused by some external causes (to the processor):
- Reset: A signal asserted on the appropriate pin;
- NMI: A rising edge of NMI signal asserted on an appropriate pin;
- Hardware Interrupts: Six hardware interrupt requests can be made via asserting signal on any of 6 external pins.

Hardware interrupts can be masked by setting appropriate bits in Status register;

Exceptions by External Causes

Hardware Interrupts: Six hardware interrupt requests can be made via asserting signal on any of 6 external pins. Hardware interrupts can be masked by setting appropriate bits in Status register.
MIPS Exceptions: A Subset (continued)

C. Exceptions that occur as result of instruction problems:
• Address Error: a reference to a nonexistent memory segment, or a reference to Kernel address space from User Mode;
• Reserved Instruction: A undefined opcode field (or privileged instruction in User mode) is executed;
• Integer Overflow: An integer instruction results in a 2’s complement overflow;
• Floating Point Error: FPU signals one of its exceptions, e.g. divide by zero, overflow, and underflow)

D. Exceptions caused by executions of special instructions:
• Syscall: A Syscall instruction executed;
• Break: A Break instruction executed;
• Trap: A condition tested by a trap instruction is true;

MIPS Exception Processing

When any of the exceptions previously listed occurs, MIPS processor processes the exception in the following 3 steps:

Step 1.
• EPC register gets a value equal to either:
  – address of a faulty instruction if the instruction itself caused exception (e.g. address error, reserved instruction) or detected hardware malfunctioning (e.g. bus error),
  – address of the next instructions which would have been executed, in all other cases.

Additionally, in the case of the address error, BadVAddr register gets value of the invalid address.

MIPS Exception Processing (continued)

Step 2. (Simplified)
• PC ← 8000018016
  – next instruction executed is at the location 8000018016
• Cause register ← a code of the exception
  – Each exception has its code, e.g.:
    • hardware interrupt = 0
    • illegal memory address (load/fatch or store) = 4 or 5
    • bus error (fetch or load/store)= 6 or 7
    • syscall instruction execution = 8
    • illegal op-code, i.e. reserved or undefined op-code= 10
    • integer overflow = 12
    • Floating point exception = 15

Step 3.
• Processor is now in Kernel mode, i.e. CPU mode bit ← 1;

Dual-Mode of CPU Operation

• CPU mode bit added to computer hardware to indicate the current CPU mode: 1 (=kernel) or 0 (=user).
• When an exception or interrupt or fault occurs CPU hardware switches to the kernel mode.

Privileged instructions can be executed only in kernel mode.
• Problem for bonus points:
  • OS loads the exception handling routine at the address 8100 0008\textsubscript{16}. What else should be done so this routine is activated each time an exception happens? Your solution should include instructions.
  • Answer by e-mail by Monday 11:00

• Problem: OS loads the exception handling routine at the address 8100 0008\textsubscript{16}. What else should be done so this routine is activated each time an exception happens? Your solution should include instructions.

• Answer: Memory location 80000180\textsubscript{16} should contain instruction \texttt{j} 400002\textsubscript{16} i.e.

\[
\text{mem location } 80000180 \rightarrow \begin{array}{cccccccc}
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & & & & & & & \\
\end{array}
\]

• Effect of \texttt{j} instruction: $\text{PC} \leftarrow [\text{PC}_{31..28}] \| [I_{25..0}] \| 0^2$

  • $\text{PC} \leftarrow 1000 0001 0000 0000 0000 0000 0000 1000$

Comments:
- \texttt{sll} instruction is noop instruction
- Add \texttt{srav}, \texttt{srlv}, \texttt{sllv}
- Mention two branch and link instructions
- Consider adding few fp instructions.