Pentium vs. Power PC
Computer Architecture and PCI Bus Interface

CSE 3322
Pentium vs. Power PC Computer Architecture and PCI Bus Interface

Nowadays, there are two major types of microprocessors in the market. The leading product is the Pentium microprocessor followed by the PowerPC microprocessor. The Pentium is product of Intel Corporation while the PowerPC is a product of IBM Corp. and Motorola Corp., and is used in Apple Macintosh computers. The basic difference between these two microprocessors is their instruction sets. Intel’s Pentium uses CISC (Complex Instruction Set Computer) technology while the PowerPC uses RISC (Reduced Instruction Set Computer) technology.

Background

Pentium is a member of the series 8086. It uses the same basic pattern as its predecessor, made 20 years ago. To keep its compatibility with the existing software base, each time it is improved, Intel’s engineers must retain its old features. On the other hand, PowerPC uses the totally new technology called RISC. In RISC, the number of instructions, addressing modes, and format are reduced. RISC technology was developed by IBM scientists in the mid-1970s, but in 1980 David Patterson of the University of California at Berkeley brought the RISC concept to the attention of computer scientists.

Instruction Size

Pentium use varies instruction set sizes. In the Pentium, instruction set format can be 1, 2, or even 6 bytes (Fig. 1). On the other hand, the PowerPC, like other RISC processors uses 4 byte instruction sets. All the instruction sets have the same length. If there are instructions with size less than 4 bytes, then they are filled in with zeros. Conversely, if there are instructions more than 4 bytes long, then they are split into instructions with a maximum size of 4 bytes. This difference of instruction sizes affects the pipelining process between the two processors.

Instruction Set

The PowerPC only has a few types of instruction sets. If there are instructions which are not in the PowerPC’s instruction set library, then the programmer/compiler has to implement those instructions by using available PowerPC instructions. That is one reason why PowerPC programmers like to use high level language rather than assembly language. Inversely, Pentium uses many instruction sets. This allows the programmers to implement their program using assembly language. It also decreases the length of program in CISC rather than in RISC.

Besides the length, programming in RISC also requires larger memory size. Fortunately, memory prices are becoming lower these days, so memory is not a major factor anymore. Although it is more difficult to program in RISC, but the fact is that 80% - 95% of the instructions are executed with only one clock cycle, in contrast to CISC instructions which require more than one clock cycle per instruction. Even the other RISC instruction
that are executed with 2 clock cycle can be executed with only one clock cycle by using
the code scheduling method. Code scheduling is the job of the compiler.
Since CISC has such a large number of instructions, each with so many different
addressing modes, microinstructions (microcode) are used to implement them. The
implementation of microinstructions inside the CPU takes more than 60% of transistors
in many CISC processors. However, in the case of RISC, due to their small set of
instructions, they are implemented using the hardwire method. Hardwiring of RISC
instructions takes no more than 10% of the transistors. It is interesting to note that in the
Pentium, a CISC processor, the V-pipe executes only simple instructions and it is
hardwired while the U-pipe executes any of 80x86 instructions and uses
microinstructions. While Pentium uses 7.5 million transistors to achieve such an
impressive performance, PowerPC processor with the same performance level can be
designed using less than 70% of Pentium transistors. There is only one problem. It will
not run the massive number software packages written for the 80x86 MS DOS PC.

Load/Share Architecture

In the PowerPC, data can only be manipulated in registers. All instructions using data
from the memory must be performed by first bringing both operands into registers inside
the CPU, then performing the arithmetic or logic operation. After that, the result is sent
back to memory. Instructions can only load operands from memory into registers or store
operands from registers into memory locations. Data must be loaded from memory into
registers, executed, and then restored again into their memory locations. There is no
direct way of doing arithmetic and logic instructions between registers and contents of
memory locations. This idea is commonly referred to as load/store architecture. In
Pentium processor, data can be manipulated while it is still in the memory.

Number of Registers

One of the major characteristics of RISC architecture is a large number of registers.
PowerPC microprocessors has 32 general purpose registers, GPR0 – GPR31, each 32 bits
wide. Of these 32 registers, only a few of them are assigned to a dedicated function. For
example, GPR0 is automatically assigned the value zero and no other value can be
written to it. One advantage of a large number of registers is that it avoids the use of the
stack to store parameters. Although a stack can be implemented on a Power PC
processor, it is not as necessary as in a Pentium processor since there are so many
registers available. It must be noted that PowerPC processors have another 32 registers
for floating-point operations, besides the 32 general-purpose registers. The floating-point
register can be configured as 64-bit in order to handle double-precision operands.

Pipelines
By increasing the working frequency, we can increase the CPU process information faster. This is technology dependent. Another way to increase CPU process information is to change the internal architecture of the CPU. An example of this technique is pipelining. Pipelining is a technology that allows the CPU to fetch and execute data at the same time.

Although many CISC microprocessors are pipelined, there exists an inherent difficulty in managing a pipeline in a system with a variety of instruction sizes and different instruction execution lengths.

Dynamically scheduled pipelines are used in Pentium and Power PC. The instruction cache fetches 16 bytes of instructions and send them to an instruction queue: four instructions for Power PC and a variable number instructions for the Pentium. Next, several instructions are fetched and decoded. Both processors use 512-entry branch history table to predict branches and speculatively run instructions after the predicted branch. Each of the instructions and all of its operands are then sent to the reservation station of one of the six functional units. An entry is placed by the dispatcher for the instruction in the reorder buffer of the commit unit. Thus an instruction cannot issue unless there is space available in both a reservation station and a reorder buffer.

Figure 2 list the specific parameters for the Power PC 604 and Pentium pipelines. Many of the differences between the Pentium and the Power PC 604 are slight. The largest difference is in the decode and dispatch. Rather than trying to pipeline variable-length 80x86 instructions, the Pentium decode unit translates the Intel instructions into 72-bit microoperations and then sends these microoperations to the reorder buffer and reservation stations. This translation takes a total of three clock cycles; one clock cycle to determine the length of the 80x86 instructions and then two more to create the microoperations.

Floating Point

The similarities of Pentium and PowerPC in floating point are both of them complying with IEEE-754 standard single-and-double precision floating point arithmetic and having regular multiply and divide instructions that operate entirely on registers. The main differences are in the floating-point instructions. Both 32 single and 32 double precision floating-point registers are offered in the Power PC.

Intel provides stack architecture with its floating-point instructions: loads push numbers onto the stack, operations find operands in the two top elements of the stacks, and stores can pop elements off the stack. Intel improved this stack architecture with instructions and addressing modes that allowed the architecture to have some of the benefits of a register-memory model. In addition to finding operands in the top two elements of the stack, one operand can be in memory or in one of the seven registers on-chip below the top of the stack. A complete stack instruction set can then be supplemented by a limited set of register-memory instructions. Another interesting feature of this architecture is that the operands in the register stack are wider than operands that are stored in memory, and all operations are performed at this wide internal precision. Unlike the maximum of 64 bits on the Power PC, the 80x86 floating point operands on the stack are 80 bits wide. Numbers are converted to the
internal 80-bit format on a load and converted back to the appropriate size on a store. Memory data can be 32-bit (single precision) or 64-bit (double precision) floating-point numbers. The register-memory version of these instructions will then convert the memory operand to this Intel 80-bit format before performing the operation. The data transfer instructions also will automatically convert 16- and 32-bit integers to floating point, and vice versa, for integer loads and stores.

**Memory Hierarchy**

Both the Pentium and Power PC offer support for secondary caches off the main CPU die. The Pentium is unique because it uses a 256-KB or 512-KB secondary cache that is a separate die integrated into the same package. This organization allows a reduced access time to the secondary cache and also reduces the number of pins from the packages, since the secondary cache pins remain inside the package. The PowerPC uses more conventional SRAMs that are separately packaged.

Both the Pentium and PowerPC have additional optimizations that allow them to reduce the miss penalty.

The PowerPC implements optimizations which allow the processor to continue executing instructions during cache misses and to resume execution as soon as the critical word is delivered back to the cache.

The Pentium goes a step further than these optimizations, allowing the processor to continue to execute instructions that access the data cache during a cache miss, as opposed to the simpler schemes that allow only instructions that do not use the data cache to be executed. This technique, called a nonblocking cache, is becoming widespread as designers attempt to hide the cache miss latency. The Pentium implements both flavors of nonblocking. Hit under miss allows additional cache hits during a miss, while miss under a miss allows multiple outstanding cache misses. The first of these two aims at hiding some of the miss latency with other work, while the second aims at overlapping the latency of two different misses.

The Pentium and PowerPC differ in their address translation, and these differences carry over into the TLB hardware. The PowerPC architecture has a much larger 52-bit virtual address versus the Pentium’s 32-bit address space. At the primary cache level, the PowerPC and Pentium differ in size and in the optimizations to reduce the miss penalty. Other than these differences, the primary caches for the two processors are very similar.

**PCI LOCAL BUSES (introduction and history)**

Just as a high-performance car needs high-performance roads (no bumps, no speed limit) to explore its full potential, high-performance CPUs also require high-performance buses. While microprocessor performance is rapidly rising, buses are not keeping up. When 80286 microprocessors of 10 – 16 MHz were used, many manufacturers resorted to proprietary buses to overcome the 8-MHz limitation associated with the ISA bus. This was especially the case where memory was concerned. In 80286/386 systems with 16 or 20 MHz speed, memory boards plugged into expansion slots could be accessed no faster than 8 MHz. This fact led manufacturers such as Compaq to have their own memory...
expansion modules. In such systems, while ISA expansion slots are used for peripheral boards such as video, hard disk, or network cards, memory expansion was done by a specially designed slot on the motherboard used only for memory modules. These memory modules work at the same speed as the CPU, or close to it. These systems were often advertised as dual-bus systems. One bus was for the ISA cards and another one was for the memory modules. In the late 1980s with the widespread adaptation of SIP (single in-line pin) and SIMM (single in-line memory module), this problem was resolved. However, the lack of a bus standard for video and other adapter cards such as disk controllers forced PC board designers to come up with what is called a local bus.

The idea of a local bus is to access the system buses at the same speed as the microprocessor, or close to it. In a 33-MHz microprocessor system with both ISA and local buses, the speed of the ISA bus signals are limited to 8 MHz, but the local bus signals are accessed at the same speed as the CPU, 33 MHz. The gap between the CPU speed and expansion slot speed started to develop when the 80286 speed exceeded 8 MHz. In those days, there were not many devices that needed speed beyond 8 MHz. This changed with the introduction of graphical user interface (GUI) software such as Microsoft Windows. In ISA bus systems, even the 16-bit video card plugged into the ISA expansion slot was not fast enough to keep up with the demand of the graphics software. With the introduction of high performance microprocessors such as the Pentium, Intel had to do something about bus performance lest their processor be buried under slow buses. For this reason Intel introduced a new local bus standard called PCI (Peripheral Component Interconnect). The first generation of the Power PC microprocessor (Power PC 601 up to Power PC 603) is not compatible to PCI Bus interface, but the next Power PC 604 is in compliant with PCI 2.0.

**PCI Local Bus**

High-performance microprocessors such as the 486 and Pentium require a high bus bandwidth to take advantage of their full potential. Therefore, it is not surprising that Intel became involved in defining a new bus standard. Although Intel came up with the specifications of the PCI local bus, it has become available free of charge to all PC and add-in board manufacturers. PCI was conceived as a specification standard for peripheral connections for Intel’s high performance microprocessors such as the 80486 and Pentium. Later, with encouragement and input from the PC industry, it has become a local bus standard with the pin-out for expansion slot connections. It has incorporated the following major characteristics: (a) burst mode data transfer, (b) level-triggered interrupts, (c) bus mastering, (d) automatic configuration, and (e) high bus bandwidth. More important, it has a bridge that allows any kind of add-in card based on ISA, EISA, or MCA to be plugged into the PCI local bus. PCI local bus characteristics are listed next.

**PCI Local Bus Characteristics**
1. It has a maximum speed of 33 MHz.
2. It has 32 and 64 bit data paths.
3. It supports burst mode data transfer of 2-1-1-1 used by microprocessors such as the 486 and Pentium.
4. It supports bus mastering, allowing the implementation of multiprocessors where any number of microprocessors can become master and control of the buses.
5. It is compatible with ISA, EISA, and MCA. With implementation of a bus bridge, it supports the slow ISA, EISA, and MCA buses. Buffers in the bridge allow the microprocessors to write into the buffers and go about its own business leaving the task to the bridge.
6. The PCI local bus is processor independent. It can be used with any microprocessor, not just Intel 80x86. For this reason, companies such as Digital Equipment and Apple have also announced support for the PCI to be used with their non-86x86 microprocessor. These features ensure that future changes in the 80x86 family will not make PCI an obsolete bus.
7. It supports 5- and 3.3-V expansion cards, allowing smooth transition from 5- to 3.3-V systems. The placing of small cutouts (keys) prevents users from plugging a card with one voltage into a motherboard with a different voltage.
8. It provides autoconfiguration capabilities, where a user can install a new add-in card without setting DIP switches, jumpers, and selecting the interrupt. Configuration software automatically selects an unused address and interrupt to resolve conflicts.
9. It has a ground or Vcc pin in between every two signals to reduce crosstalk and radio frequency emissions.
10. It implements level-triggered interrupts, which support interrupt sharing.
11. It supports up to 10 peripherals. Some of the peripherals must be embedded into the motherboard.
12. The maximum number of expansion slots working at 33 MHz varies, depending on the 5-V or the 3.3-V implementation. An increase in the number of expansion slots beyond 5 means a speed lower than 33 MHz. The use of a highly refined connector with a small area of contact makes the PCI bus a high-frequency bus.

### PCI

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus type</td>
<td>Backplane</td>
</tr>
<tr>
<td>Basic data bus width (signals)</td>
<td>32-64</td>
</tr>
<tr>
<td>Address/data multiplexed</td>
<td>Multiplexed</td>
</tr>
<tr>
<td>Number of bus masters</td>
<td>Multiple</td>
</tr>
<tr>
<td>Arbitration</td>
<td>Centralized, parallel arbitration</td>
</tr>
<tr>
<td>Clocking</td>
<td>Synchronous 33-66 MHz</td>
</tr>
<tr>
<td>Theoretical peak bandwidth</td>
<td>133-512 MB/sec</td>
</tr>
<tr>
<td>Estimated typical achievable bandwidth for basic bus</td>
<td>80 MB/sec</td>
</tr>
<tr>
<td>Maximum number of devices</td>
<td>1024 (with multiple bus segments; at most 32 devices/bus segment)</td>
</tr>
<tr>
<td>Maximum bus length</td>
<td>0.5 meter</td>
</tr>
</tbody>
</table>
**Figure 1:** Typical 80x86 instruction format

a. JE EIP + displacement

```
   4  4  8
JE   Cond  Displacement
```

b. CALL

```
   8        32
CALL      Offset
```

c. MOV EBX, (EDI + 45)

```
   6  1  1  8  8
MOV   d   w  r/m postbyte  Displacement
```

d. PUSH ESI

```
   5  3
PUSH  Reg
```

e. ADD EAX, #6765

```
   4  3  1
ADD  Reg  w  Immediate
```

f. TEST EDX, #42

```
   7  1  8
TEST  w  Postbyte  Immediate
```
<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Power PC</th>
<th>Pentium</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max number of instructions issued per clock cycle</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Max number of instructions completing execution per clock cycle</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>Max number of instructions committed per clock cycle</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>Number of bytes in instruction queue</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Number of instructions in reorder buffer</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>Number of entries in branch table buffer</td>
<td>16</td>
<td>40</td>
</tr>
<tr>
<td>Number of history bits per entry in branch history buffer</td>
<td>512</td>
<td>512</td>
</tr>
<tr>
<td>Number of rename buffers</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Total number of reservation stations</td>
<td>12 integer + 8 FP</td>
<td>40</td>
</tr>
<tr>
<td>Total number of functional units</td>
<td>12</td>
<td>20</td>
</tr>
<tr>
<td>Number of integer functional units</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Number of integer functional units</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Number of complex integer operation functional units</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Number of floating-point functional units</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Number of branch functional units</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Number of memory functional units</td>
<td>1 for both load and store</td>
<td>1 for load + 1 for store</td>
</tr>
</tbody>
</table>
References


