Structure of the Pentium Microprocessor

Youwei Lu

CISC 530-51- A-2018/Late Fall Assignment 2

Introduction

Pentium is a brand used for a series of x86 architecture-compatible microprocessors produced by Intel since 1993. Pentium processors are considered entry-level products that Intel rates as "two stars" (Hesseldahl, 2000), meaning that they are above the low-end Atom and Celeron series, but below the faster Core i3, i5, i7, i9, and high-end Xeon series.

Pentium Family

Intel first introduced microprocessors in 1969, with a 4-bit microprocessor 4004, then 8-bit microprocessors 8080 and 8085. Soon, 16-bit processors 8086 was introduced in 1979, which has 20-bit address bus and 16-bit data bus. 8088 is a less expensive version, which uses 8-bit data bus. Since then, the 80 series developed rapidly. 80186 in 1979, and 80286 was introduced in 1982, with 24-bit address bus, 16MB address space, enhanced with memory protection capabilities, and a protected mode was introduced. In 1985, the first 32-bit processor was invented: 80386. It supports 32-bit data bus and 32-bit address bus, 4GB address space, and introduced paging. An improved version 80486 was introduced in 1989.

The time comes to 1993, and a new era arrived. Following their earlier naming conventions, Intel's new fifth-generation chip was expected to be named the 586. However, Intel wanted to be able to register as a trademark the name of their new processor, and since numbers cannot be trademarked, the Pentium was born. Since this time the Pentium name has become one of the most widely recognized trademarks throughout the computer world. Intel announced its next generation of compatible microprocessors following the popular i486 CPU family. The design started in early 1989 with the primary goal of maximizing performance while preserving software compatibility within the practical constraints of available technology. The Pentium processor integrates 3.1 million transistors and carries the Intel trademark. (Alpert & Avnon, 1993).

Each processor in the Pentium series incorporates and builds off of the previous processor's architectural achievements. (Dandamudi, 2003) Pentium (80586) is similar to 486 but with a 64-bit data bus, and wider internal datapaths (128- and

256-bit wide), second execution pipeline added (to make superscalar performance and two instructions/clock), and doubled on-chip L1 cache. A branch prediction was also added. In 1995, Pentium Pro was introduced to carry three-way superscalar (3 instructions/clock), 36-bit address bus (so total is 64GB address space), and dynamic execution. In addition to the L1 cache, it has 256 KB L2 cache. With the commercial success of Pentium, Pentium II was introduced in 1997, and Pentium III in 1999, Pentium 4 in 2000. After that, Intel decides to use another brand name to make new generation microprocessors. For example, Itanium processor was the first one to use RISC design, instead of CISC.

Pentium Overview

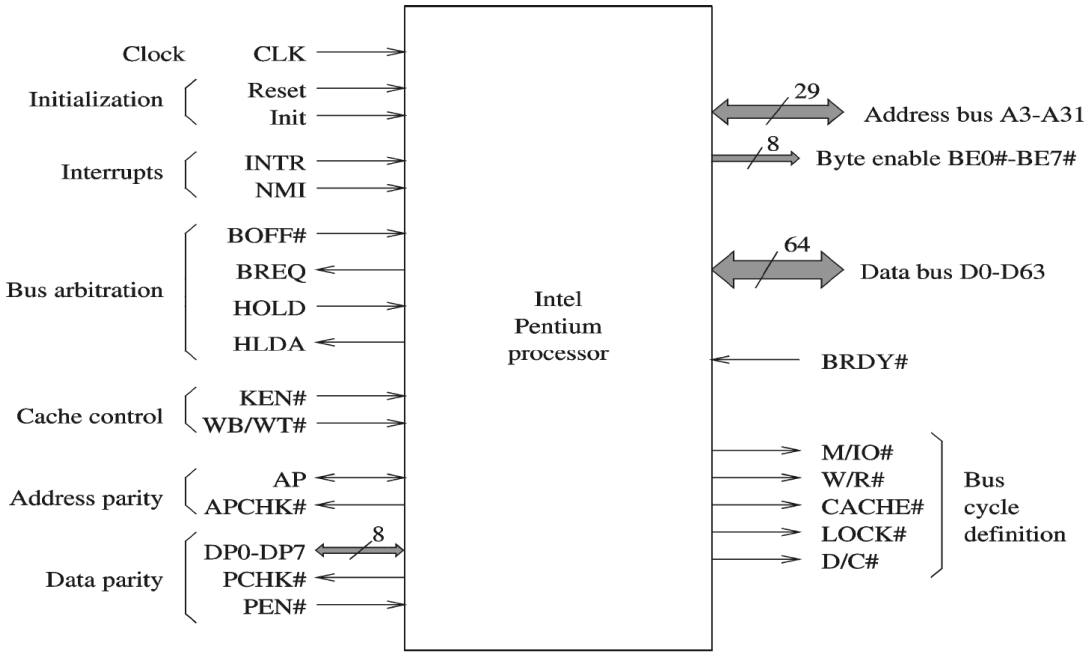


Figure 1. A typical Pentium layout. (Dandamudi, 2003)

A typical Pentium layout is shown in Figure 1, and their addresses are explained below:

- Data bus (D0 - D63): 64-bit data bus.
- Address bus (A3 - A31): Only 29 lines, no A0-A2 (due to 8-byte wide data bus).
- Byte enables (BE0# - BE7#): Identifies the set of bytes to read or write. BE0# specifies the least significant byte (D0 - D7), BE1# specifies the next byte (D8 - D15), ... , BE7# specifies the most significant byte (D56 - D63). Any combination of bytes can be specified.

- Data parity (DP0 - DP7): Even parity for 8 bytes of data. DP0 for D0 - D7, DP1 for D8 - D15, ... , DP7 for D56 - D63.
- Parity check (PCHK#): Indicates the parity check result on data read. Parity is checked only for valid bytes, indicated by BE# signals.
- Parity enables (PEN#): Determines whether the parity check should be used.
- Address parity (AP): Bad address parity during inquire cycles.
- Memory/IO (M/IO#): Defines bus cycle: memory or I/O.
- Write/Read (W/R#): Distinguishes between write and read cycles.
- Data/Code (D/C#): Distinguishes between data and code.
- Cacheability (CACHE#): Read cycle: indicates internal cacheability. Write cycle: burst write-back.
- Bus lock (LOCK#): Used in a read-modify-write cycle. Useful in implementing semaphores.
- Interrupt (INTR): External interrupt signal.
- Nonmaskable interrupt (NMI): External NMI signal.
- Clock (CLK): System clock signal.
- Bus ready (BRDY#): Used to extend the bus cycle. Introduces wait states.
- Bus request (BREQ): Used in bus arbitration.
- Backoff (BOFF#): Aborts all pending bus cycles and floats the bus, useful to resolve the deadlock between two bus masters.
- Bus hold (HOLD): Completes outstanding bus cycles and floats bus. Asserts HLDA to give control of the bus to another master
- Bus hold acknowledges (HLDA): Indicates the Pentium has given control to another local master. Pentium continues execution from its internal caches.
- Cache enables (KEN#): If asserted, the current cycle is transformed into cache line fill.
- Write-back/Write-through (WB/WT#): Determines the cache write policy to be used.
- Reset (RESET): Resets the processor, starts execution at FFFFFFF0H. Invalidates all internal caches.

- Initialization (INIT): Similar to RESET but internal caches and FP registers are not flushed. After powerup, use RESET (not INIT)

Pentium Structure

The Pentium processor is the successor to the Intel486 processor. Originally released with a 66MHz clock speed, it is a 16-bit based superscalar processor capable of executing two instructions in parallel during a single clock. It uses a CISC (Complex Instruction Set Computer) type instruction set and uses the little-endian type format to store bytes in memory. A 64-bit external bus, separate data, and instruction caches write buffers, and a pipelined floating-point unit combine to sustain a high executing rate. Caching along with pipeline and instruction flow are discussed in detail in this article.

A Pentium processor's major functional components are: (pctechguide, 2018)

- Core: The heart of a Pentium is the execution unit. The Pentium has two parallel integer pipelines enabling it to read, interpret, execute and despatch two instructions simultaneously.

- Branch Predictor: The branch prediction unit tries to guess which sequence will be executed each time the program contains a conditional jump, so that the Prefetch and Decode Unit can get the instructions ready in advance.

- Floating Point Unit: The third execution unit in a Pentium, where non-integer calculations are performed.

- Level 1 Cache: The Pentium has two on-chip caches of 8KB each, one for code and one for data, which are far quicker than the larger external secondary cache.

- Bus Interface: This brings a mixture of code and data into the CPU, separates the two ready for use, and then recombines them and sends them back out.

All the elements of the processor stay in step by use of a clock which dictates how fast it operates. The very first microprocessor had a 100KHz clock, whereas the Pentium Pro uses a 200MHz clock, which is to say it ticks 200 million times per second. As the clock ticks, various things happen. The Program Counter (PC) is an internal memory location which contains the address of the next instruction to be executed. When the time comes for it to be executed, the Control Unit transfers the instruction from memory into its Instruction Register (IR). (pctechguide, 2018)

At the same time, the PC is incremented so that it points to the next instruction in sequence; now the processor executes the instruction in the IR. Some instructions are handled by the Control Unit itself, so if the instruction says jump to location 2749, the value of 2749 is written to the PC so that the processor executes that instruction next. (pctechguide, 2018)

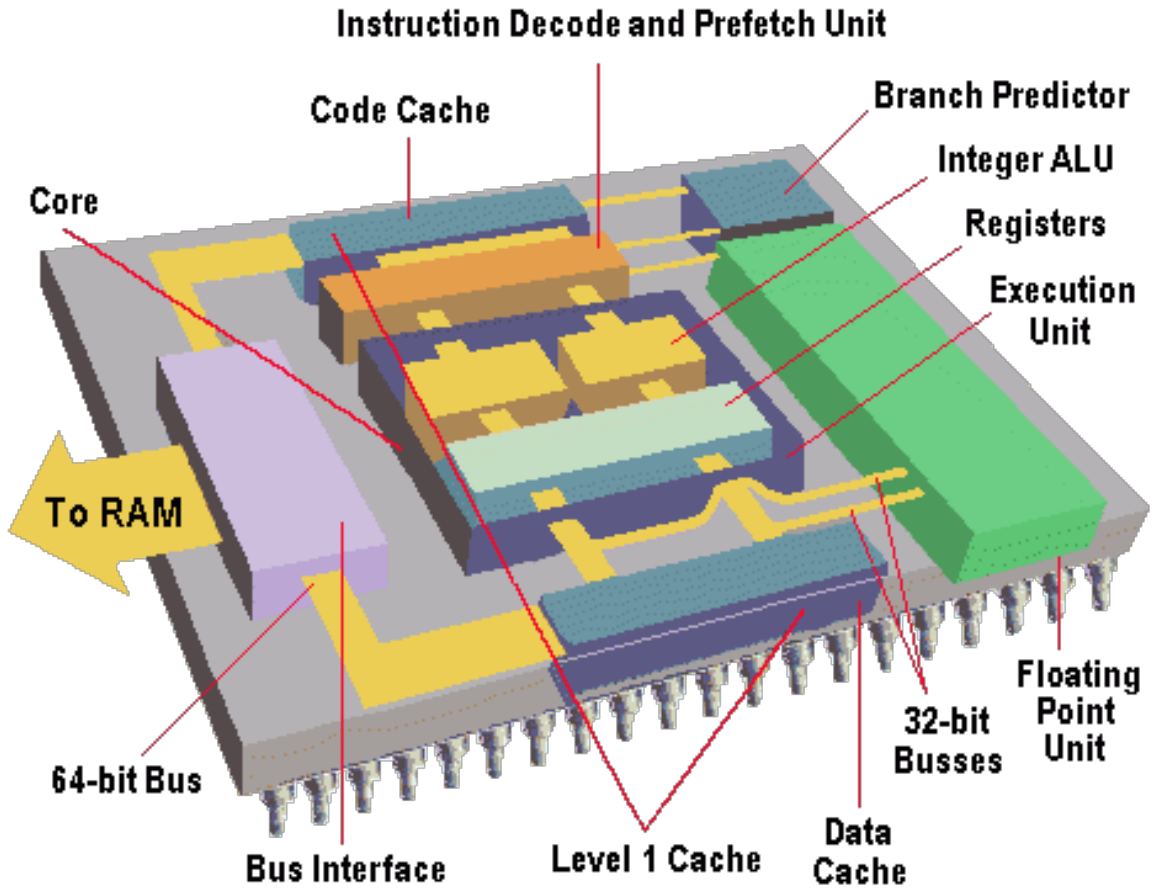


Figure 2. Basic Structure of a Pentium Microprocessor. (pctechguide, 2018)

Many instructions involve the arithmetic and logic unit (ALU). This works in conjunction with the General Purpose Registers – temporary storage areas which can be loaded from memory or written to memory. A typical ALU instruction might be to add the contents of a memory location to a general purpose register. The ALU also alters the bits in the Status Register (SR) as each instruction is executed; this holds information on the result of the previous instruction. Typically, the SR has bits to indicate a zero result, an overflow, a carry and so forth. The control unit uses the information in the SR to execute conditional instructions such as jump to address 7410 if the previous instruction overflowed. (pctechguide, 2018)

The following of this paper will introduce some basic topics about microprocessors, with special emphasize on Pentium. The concepts of basic of computer architecture will first be introduced on page 7, then the branch prediction is introduced on page 11, and the system architecture is discussed on page 16. Some advanced technologies should be studied, as rapid execution module on page 18, memory subsystem on page 19, and hyperthreading technology on page 21. As we know, Pentium uses CISC, not RISC. Their differences are introduced and compared on page 23.

As a summary and page reference, a complete explanation is given for the following issues:

- Concepts of Computer Architecture. (page 7)
- Branch Prediction. (page 11)
- System Architecture. (page 16)
- Rapid Execution Module. (page 18)
- Memory Subsystem. (page 19)
- Hyperthreading Technology. (page 21)
- RISC and CISC Convergence, Advantages of RISC, Design Issues of RISC Processors. (page 23)

Concepts of Computer Architecture

In a programmer's view, the computer architecture is a structure of a computer that a machine language programmer must understand to write a correct (timing independent) program for the machine. The computer architecture is a set of rules that explain the functionality, organization, and implementation of computer systems. (Shrestha & Ram, 2014)

One concept that people often get confused is the computer organization. It is actually from the implementer's view, which refers to the actual hardware structure and realization. Table. 1 gives a comparison of Computer Architecture and Computer Organization.

Computer Architecture	Computer Organization
Computer architecture is concerned with the structure and behavior of computer systems as seen by the user.	computer organization is concerned with the way the hardware components operate and the way they are connected together to form a computer system.
It includes information, formats, instruction set, and techniques for addressing memory.	It includes hardware details transparent to the programmer such as control signal and peripheral.
It describes what the computer does.	It describes how the computer performs. Ex, circuit design, control signals, memory types and etc.

Table 1
Computer Architecture vs Computer Organization. (Shrestha & Ram, 2014)

There are usually 6 layers of computer architecture, as shown in Figure. 3 on the following page: Each level is an abstraction of the level below it. The machines at each level execute their own particular instructions, calling upon machines at lower levels to perform tasks as required. All the work is ultimately carried out by computer circuits. (Null & Lobur, 2012).

In the first computer, ENIAC, all programming was done at the digital logic level, and a different hardware configuration was needed to solve every unique problem type. John von Neumann invented the computer that could store instructions in memory. These stored-program computers have become known as von Neumann Architecture systems.

Today's stored-program computers have the following characteristics as shown in Figure. 4 on page 9:

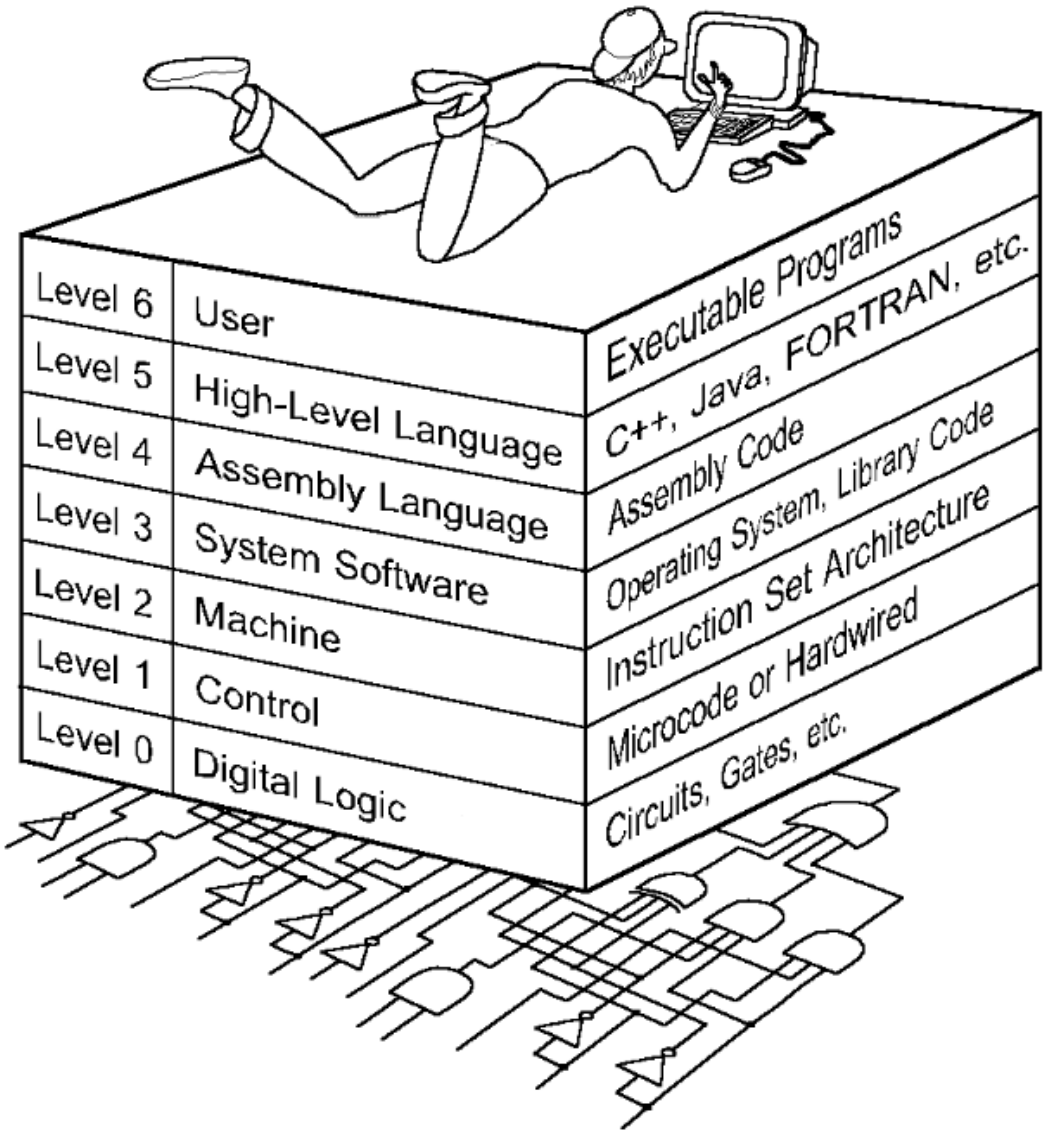


Figure 3. Computer Architecture Layers (Null & Lobur, 2012)

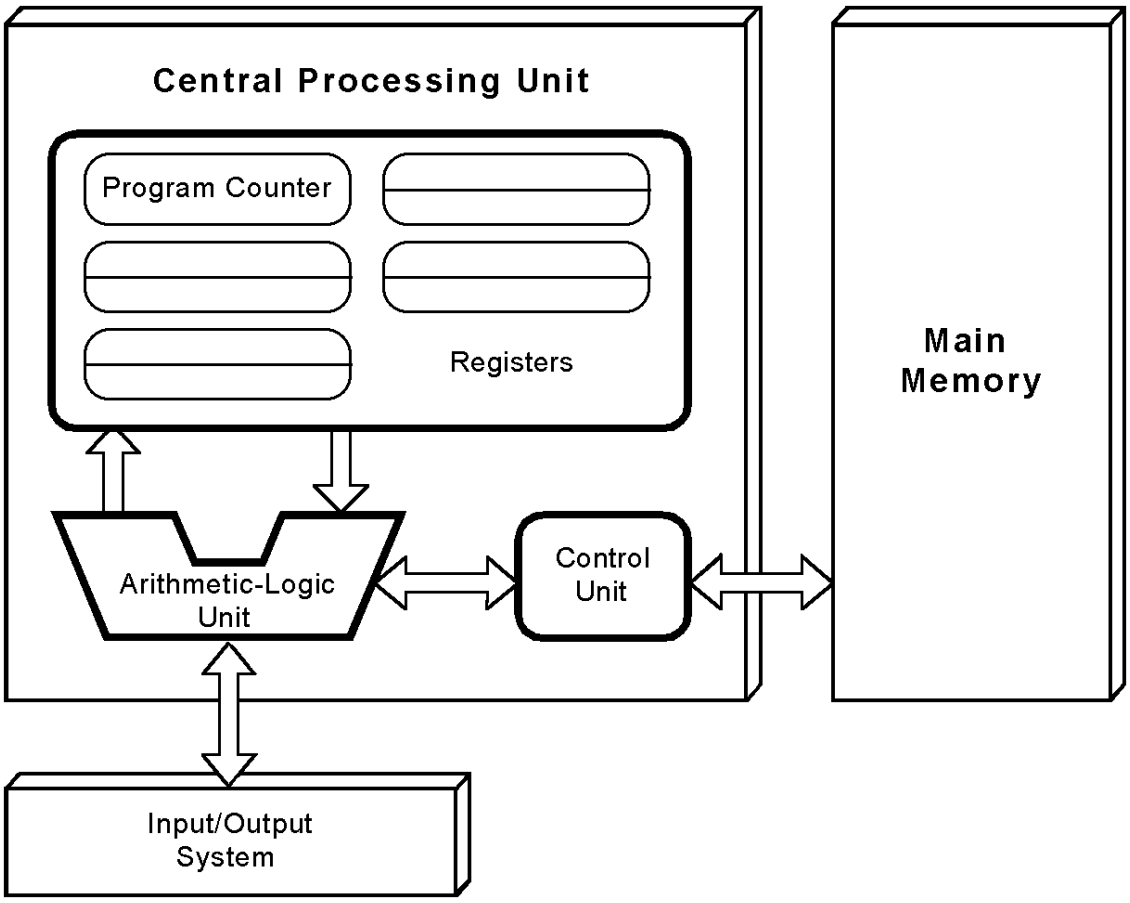


Figure 4. The von Neumann Model (Null & Lobur, 2012)

- Three hardware systems: A central processing unit (CPU), The main memory system, and An i/O system
- the capacity to carry out sequential instruction processing
- A single data path between the CPU and main memory (known as von Neumann bottleneck)

In a von Neumann model, digital computation in ALU Internal storage of data is programmable via a standard set of instructions, with internal storage of program input/output. It has automatic sequencing of instruction execution (by control mechanism).

Pentium Improvements on von Neumann

Conventional stored-program von Neumann computers have undergone many incremental improvements over the years, especially with the help from Intel. These improvements include adding processors, specialized buses, floating-point units, and

cache memories. Adding processors is a typical and most significant way to improve von Neumann architecture, and Pentium started to manufacture multi-core processors since Pentium D.

The Pentium D brand refers to two series of desktop dual-core 64-bit x86-64 microprocessors with the NetBurst microarchitecture, which is the dual-core variant of Pentium 4 "Prescott" manufactured by Intel. Each CPU comprised two dies, each containing a single core, residing next to each other on a multi-chip module package. The brand's first processor, codenamed Smithfield, was released by Intel on May 25, 2005. (Intel, 2007).

Branch Prediction

Branch prediction is one of the ancient performance improving techniques which still finds relevance to modern architectures. To understand the motivation for branch prediction, we first need to know how instruction is executed.

Each instruction includes fetch, decode, execute, and write-back stages. When waiting in a branch (e.g. an if-then-else structure), the pipeline is waiting for the result for the next step. Figure. 5 shows an example of a 4-stage pipeline. The colored boxes represent instructions independent of each other. Modern processors may have more than 10 pipeline stages between next PC calculation and branch resolution, and you can imagine how much work is lost if the pipeline doesn't follow correct instruction flow.

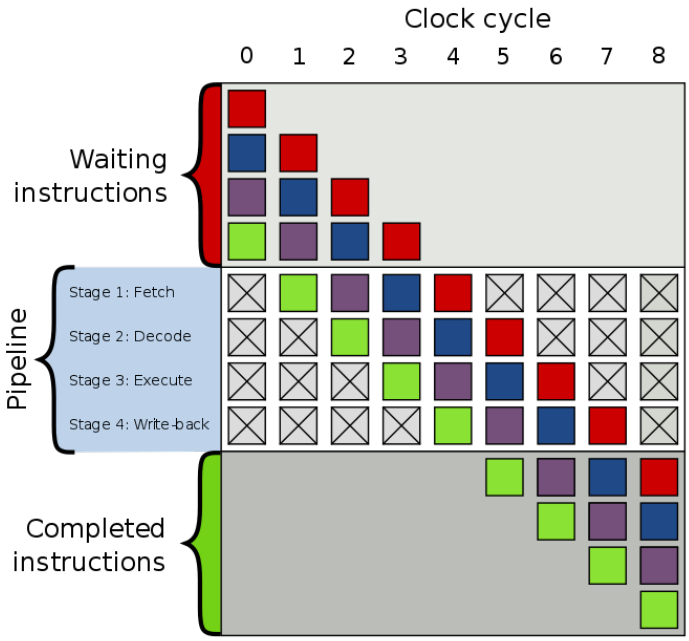


Figure 5. Example of 4-stage pipeline. (Kolinko & Lee, n.d.)

Therefore, as branch penalties limit the performance of deeply pipelined processors, modern branch predictors are used to providing high accuracy and can reduce branch penalties.

Without branch prediction, the processor would have to wait until the conditional jump instruction has passed the execute stage before the next instruction can enter the fetch stage in the pipeline. Prediction works by executing instructions from both paths of the branch and only permitting those instructions from the taken path to modify architectural state. (Vinyard, 2000)

(Vinyard, 2000) also presents a very good example as below:

Given the following source code:

```
if (emp_status == ACTIVE) {
n_active_emps++;
total_payroll += emp_pay;
}
else {
n_inactive_emps++;
}
```

This code is typically complied into a sequence such as:

```
{
cmp.ne p1 = rs, ACTIVE // compare emp_status
(p1) br else           // jump to else code if condition fails
}
.label then
{
add  rt = rt, rp       // sum total_payroll + emp_pay
add  ra = ra, 1        // increment n_active_emps
br  join
}
.label else
{
add  ri = ri, 1        // increment n_inactive_emps
}
.label join
```

The generated predicted code would look like:

```
{
cmp.eq p1, p2 = rs, ACTIVE // compare emp_status
}
{
(p1)  add rt = rt, rp      // sum total_payroll + emp_pay
(p1)  add ra = ra, 1       // increment n_active_emps
(p2)  add ri = ri, 1       // increment n_inactive_emps
}
```

In the above example, it is very clear that the three predicted instructions can be executed in parallel. Each of the instructions above is predicted. If the instruction's predicate evaluates to a 1, the instruction is executed. Otherwise, the instruction equates to a NOP. On a machine with three or more additional units, the above example utilizes the same cycles as a non-predicated machine except there is no possibility of a branch penalty. (Vinyard, 2000)

An even more clear diagram can be found in (Byte, n.d.) as shown in Figure. 6 on the next page.

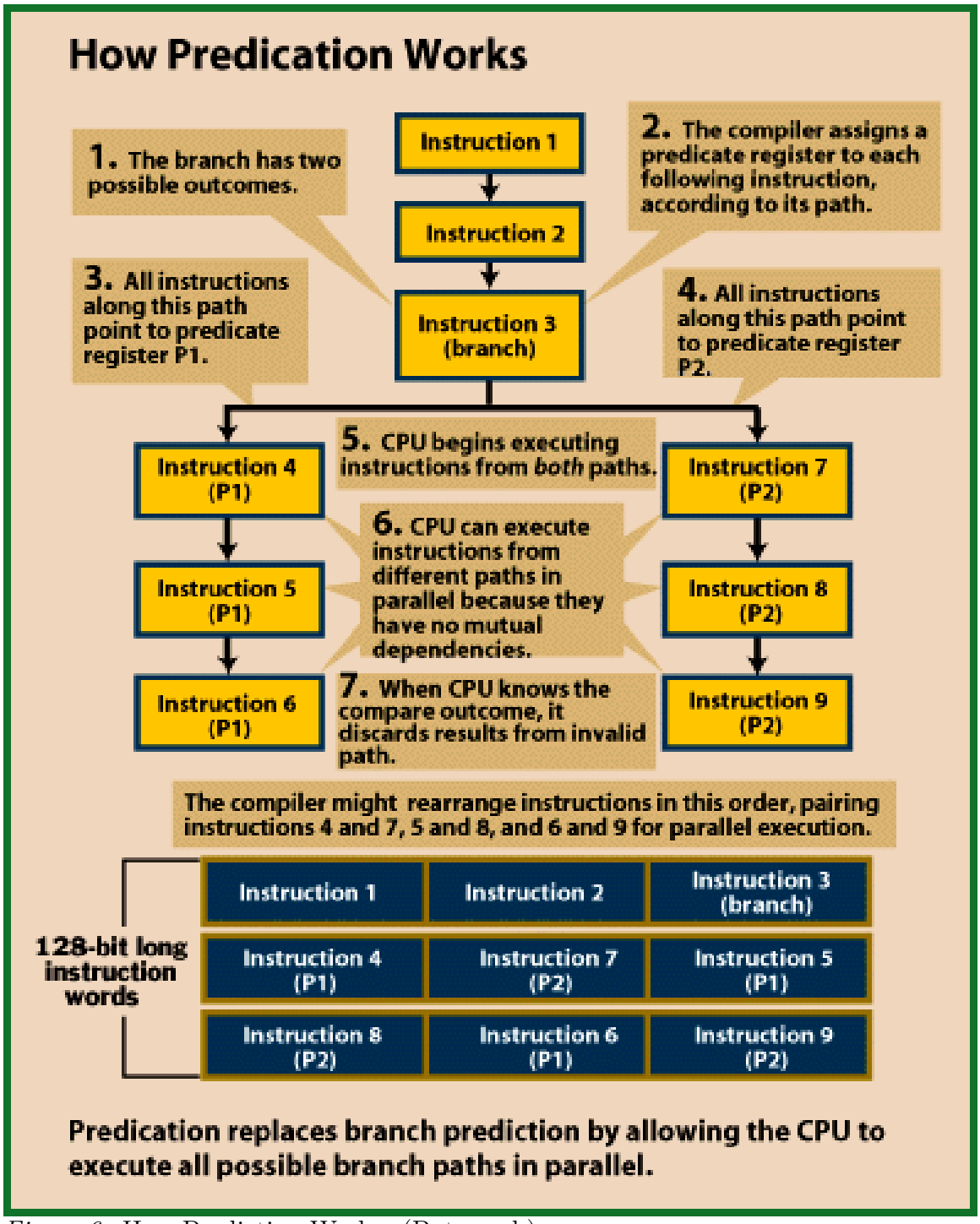


Figure 6. How Prediction Works. (Byte, n.d.)

There are typically two kinds of branch prediction: Static Branch Prediction and Dynamic Branch Prediction.

Static Branch Prediction

Static Branch Prediction predicts always the same direction for the same branch during the whole program execution. It comprises hardware-fixed prediction and compiler-directed prediction. Simple hardware-fixed direction mechanisms can be:

- Predict always not taken
- Predict always taken
- Backward branch predict taken, forward branch predict not taken

Sometimes a bit in the branch opcode allows the compiler to decide the prediction direction. (McFarland, 2006)

Dynamic Branch Prediction

Dynamic Branch Prediction: the hardware influences the prediction while execution proceeds. Prediction is decided on the computation history of the program. During the start-up phase of the program execution, where a static branch prediction might be effective, the history information is gathered and dynamic branch prediction gets effective. In general, dynamic branch prediction gives better results than static branch prediction, but at the cost of increased hardware complexity. (McFarland, 2006)

Usually, Dynamic Branch Prediction improve performance and reduce energy by decreasing the number of instructions executed on the wrong path. However, reducing latency and storage overhead of BP while maintaining high accuracy presents significant challenges. A survey of dynamic branch prediction techniques can be found in (Mittal, 2016).

Pentium on Branch Prediction

Pentium uses Dynamic Branch Prediction.

It is implemented using 4 way set associated cache with 256 entries. This is called Branch Target Buffer (BTB). The directory entry for each line consists of:

- Valid bit: indicates whether the entry is valid or not.
- History bit: Track how often bit has been taken.

Their four states can be illustrated in the diagram Figure. 7 on the following page and explained by the table:

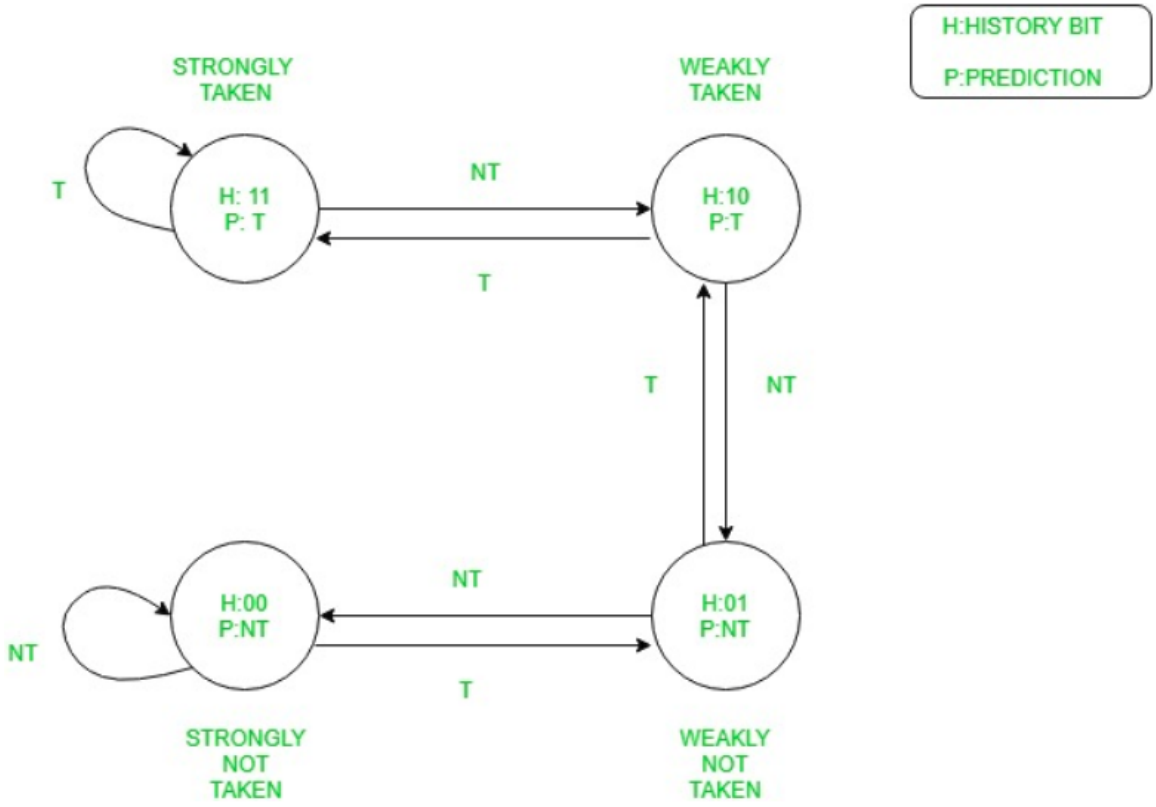


Figure 7. 4 Stage way. (GeeksforGeeks, n.d.)

History Bits	Resulting Description	Prediction Made	If Branch Taken	If Branch Not Taken
11	Strongly Taken	Branch Taken	Remains in same state	Downgraded to weakly taken
10	Weakly Taken	Branch Taken	Upgraded to strongly taken	Downgraded to weakly not taken
01	Weakly Not Taken	Branch Not Taken	Upgraded to weakly taken	Downgraded to strongly not taken
00	Strongly Not Taken	Branch Not Taken	Upgraded to weakly	not taken Remains in same state

Table 2
4 Stage way.

System Architecture

The system architecture was partly discussed in the section on page 7, and we already understand the architecture of a von Neumann computer, as clearly shown in Fig. 4 on page 9, I'd like to take the chance to focus on Pentium structure in this section.

Pentium Architecture

The first Pentium processors were introduced in 1993. It runs at a clock frequency of either 60 or 66 MHz and has 3.1 million transistors. Some of the features of Pentium architecture are (Das, n.d.)

- Complex Instruction Set Computer (CISC) architecture with Reduced Instruction Set Computer (RISC) performance.
- 64-Bit Bus.
- Upward code compatibility.
- Pentium processor uses Superscalar architecture and hence can issue multiple instructions per cycle.
- Multiple Instruction Issue (MII) capability.
- Pentium processor executes instructions in five stages. This staging, or pipelining, allows the processor to overlap multiple instructions so that it takes less time to execute two instructions in a row.
- The Pentium processor fetches the branch target instruction before it executes the branch instruction.
- The Pentium processor has two separate 8-kilobyte (KB) caches on a chip, one for instructions and one for data. It allows the Pentium processor to fetch data and instructions from the cache simultaneously.
- When data is modified, only the data in the cache is changed. Memory data is changed only when the Pentium processor replaces the modified data in the cache with a different set of data.
- The Pentium processor has been optimized to run critical instructions in fewer clock cycles than the 80486 processor.

The Pentium processor has two primary operating modes: (Das, n.d.)

- Protected Mode - In this mode all instructions and architectural features are available, providing the highest performance and capability. This is the recommended mode that all new applications and operating systems should target.

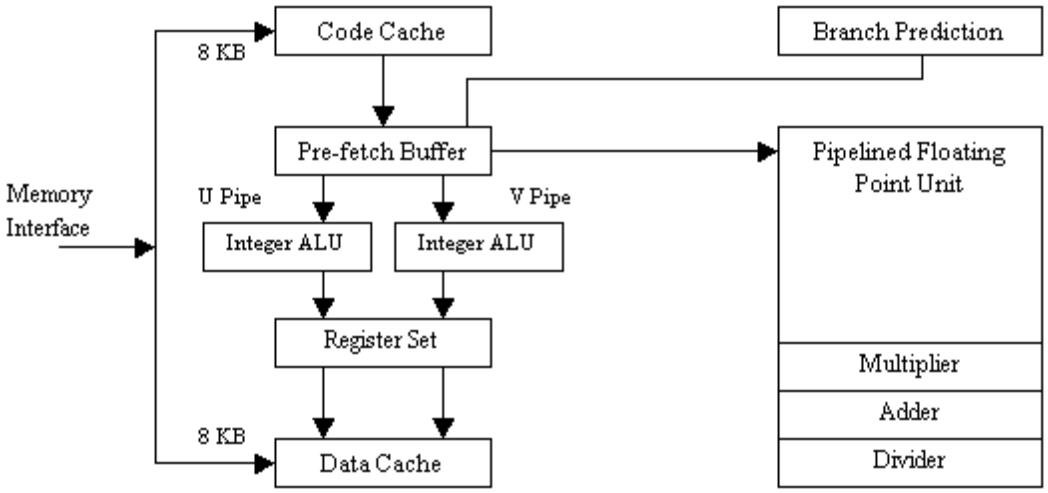


Figure 8. Superscalar Architecture of Pentium. (Das, n.d.)

- Real-Address Mode - This mode provides the programming environment of the Intel 8086 processor, with a few extensions. Reset initialization places the processor in real mode where, with a single instruction, it can switch to protected mode.

Rapid Execution Module

The Rapid Execution Module was introduced to the market by Intel and given a more beautiful commercial name: NetBURST. Using Rapid Execution Engine certain instructions may be executed at twice the normal speed.

A pipeline is an execution unit which takes in decoded micro-instructions. The X86 instructions are first decoded and then sent to the pipeline. The longer the pipeline is, the quicker an instruction can be executed. Each stage executes a minor part of the work and by spreading the work on "more hands", the efficiency is increased.(Karbosguide, n.d.)

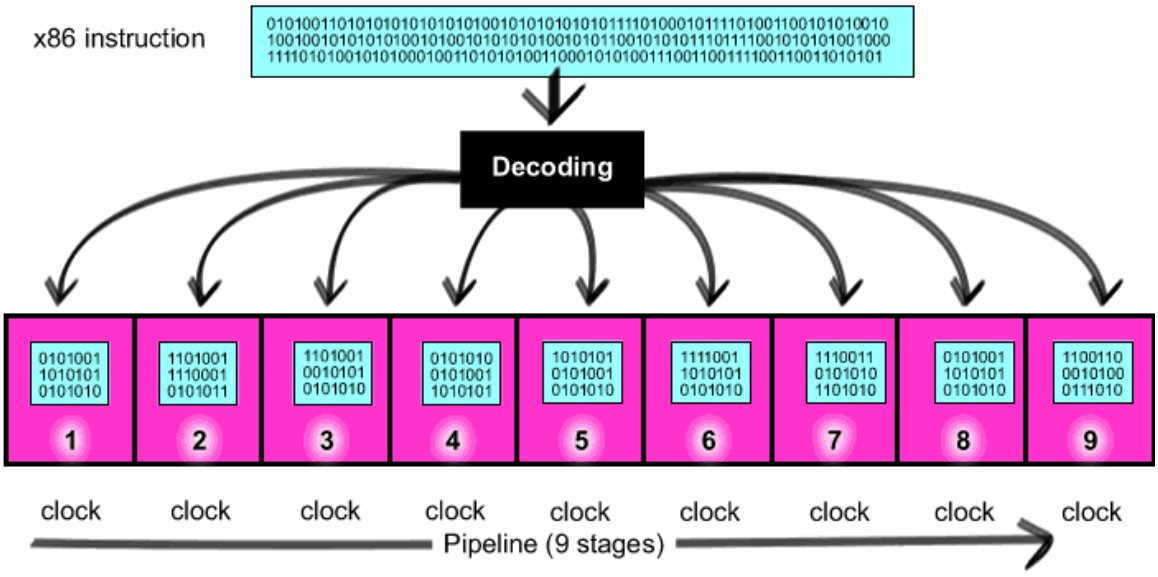


Figure 9. Stages of pipeline. (Karbosguide, n.d.)

In Pentium III the pipeline was of 10 stages. In Pentium 4 it has been increased to 20 stages.

The problem with the long pipeline is, that it takes to a long time load the instruction. And if the instruction should not have been loaded at all, the error is most costly (in time) the longer the pipeline becomes. (Karbosguide, n.d.)

With many instructions being executed simultaneously you cannot avoid loading the wrong instruction (called misprediction) from time to time. And a shorter pipeline is quicker to recover this error - fewer stages have to be cleared and reloaded. (Karbosguide, n.d.)

The benefit from a pipeline increased from 10 stages to 20, is to open up for new higher clock frequencies. When each instruction is executed in more stages, it can be done a lot quicker. (Karbosguide, n.d.)

Memory Subsystem

In a user(or even programmer)'s view, the computer memory is flat, large storage space. However, this is not possible with current technology. We wanna the fastest computer, so we can make fast memories by using static RAM, but its speed drops as the memory size increases. At the same time, Dynamic RAM can be used for slow but large memory space.

Generally, fast storage technologies cost more per byte, have less capacity, and require more power (heat!). To balance performance and cost, computer memory systems typically consist of a four-level hierarchy: Registers — Cache — Main Memory — Backing Store (i.e. disk). Note that data is moved between the cache, main memory, and backing store transparently, but data movement between registers and the rest of memory is explicitly under the control of the program. It is up to the compiler to decide which data items should be moved to registers, and when, and the compiler must compile into the code the relevant load and store instructions. (Stevens & Viglas, 2009)

In practice, many computers introduce another level into the hierarchy: they have two levels of cache; a small (perhaps 64Kbyte), very fast cache on the processor chip (called the first level cache), and a larger (perhaps 512Kbyte) second level cache, either on the processor chip or on separate chips, and intermediate in speed between the on-chip cache and the main memory. (Stevens & Viglas, 2009)

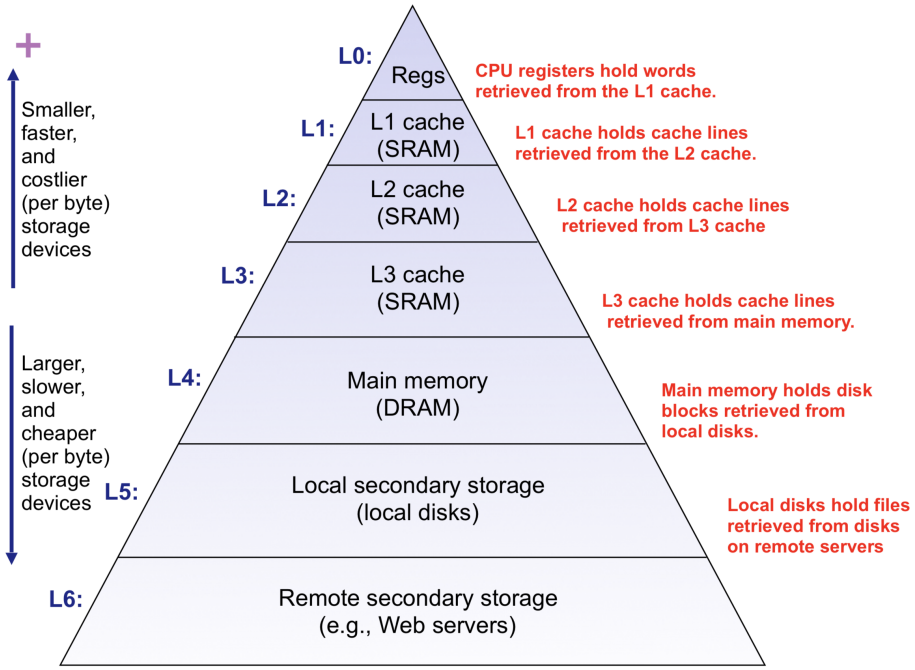


Figure 10. Memory Hierarchy (Bryant & O'Hallaron, 2015)

One can refer to Figure. 10 for a complete list of the memory hierarchy.

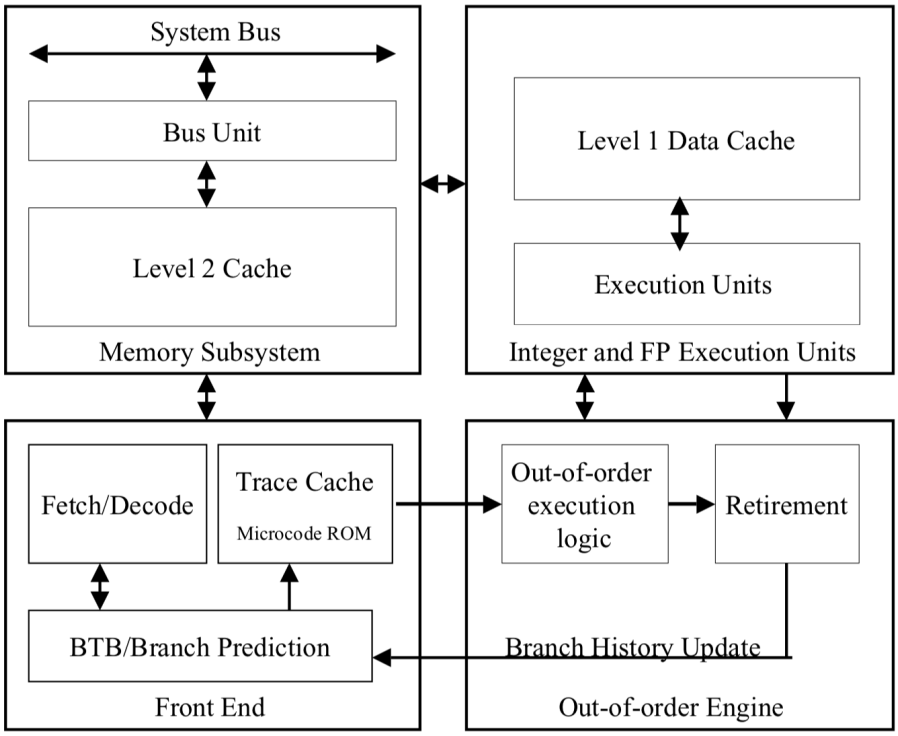


Figure 11. Pentium Memory Hierarchy (Hinton et al., 2001)

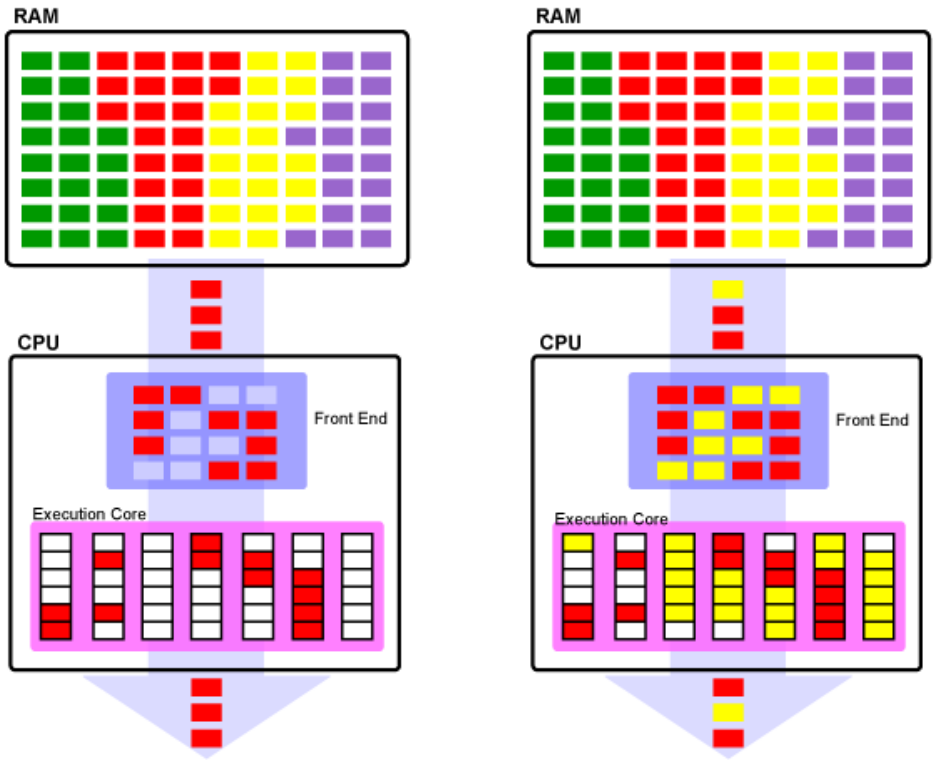
Pentium Memory Subsystem

Figure. 11 shows the memory subsystem of Pentium. This includes the L2 cache and the system bus. The L2 cache stores both instructions and data that cannot fit in the Execution Trace Cache and the L1 data cache. The external system bus is connected to the backside of the second-level cache and is used to access main memory when the L2 cache has a cache miss, and to access the system I/O resources. (Hinton et al., 2001)

Hyperthreading Technology

Hyperthreading technology is a process that allows an Intel Pentium 4 CPU to be viewed by the operating systems as two processors. From a software or architecture perspective, this means operating systems and user programs can schedule processes or threads to logical processors as they would on multiple physical processors. From a microarchitecture perspective, this means that instructions from both logical processors will persist and execute simultaneously on shared execution resources. (Marr et al., 2002)

To make the physical processor appear as multiple logical processors, there is one copy of the architecture state for each logical processor, and the logical processors share a single set of physical execution resources. From a software or architecture perspective, this means operating systems and user programs can schedule processes or threads to logical processors as they would on conventional physical processors in a multiprocessor system. From a microarchitecture perspective, this means that instructions from logical processors will persist and execute simultaneously on shared execution resources. (Marr et al., 2002)



(a) Single-threaded CPU. (Stokes, 2002) (b) Hyper-threaded CPU. (Stokes, 2002)
Figure 12. Comparison of single and hyperthreading CPU

To clearly show the improvement, we may compare the front-end behavior in

CPU. The front end of the pipeline is responsible for delivering instructions to the later pipe stages.

In Figure. 12a on the preceding page, the different colored boxes in RAM represent instructions for four different running programs. As you can see, only the instructions for the red program are actually being executed right now. (Stokes, 2002)

In comparison, Figure. 12b on the previous page illustrates that super-threading has all instructions issued by the front end on each clock from the same thread.

RISC and CISC Convergence, Advantages of RISC, Design Issues of RISC Processors

CISC stands for "Complex Instruction Set Computer", and RISC stands for "Reduced Instruction Set Computer". CISC systems try to improve performance by reducing the number of instructions per program. In contrary, RISC system shortens execution time by reducing the clock cycles per instruction.

RISC

Advantages of RISC processor architecture: (elprocus, n.d.)

- Because of the small set of instructions of RISC, high-level language compilers can produce more efficient code.
- RISC allows freedom of using the space on microprocessors because of its simplicity. Instead of using Stack, many RISC processors use the registers for passing arguments and holding the local variables.
- RISC functions use only a few parameters, and the RISC processors cannot use the call instructions, and therefore, use fixed length instructions which are easy to pipeline.
- The speed of the operation can be maximized and the execution time can be minimized.
- Very less number of instruction formats (less than four), a few numbers of instructions (around 150) and a few addressing modes (less than four) are needed.

Drawbacks of RISC processor architecture: (elprocus, n.d.)

- With the increase in the length of the instructions, the complexity increases for the RISC processors to execute due to its character cycle per instruction.
- The performance of the RISC processors depends mostly on the compiler or programmer as the knowledge of the compiler plays a major role while converting the CISC code to a RISC code; hence, the quality of the generated code depends on the compiler.
- While rescheduling the CISC code to a RISC code, termed as a code expansion, will increase the size. And, the quality of this code expansion will again depend on the compiler, and also on the machine's instruction set.
- The first level cache of the RISC processors is also a disadvantage of the RISC, in which these processors have large memory caches on the chip itself. For feeding the instructions, they require very fast memory systems.

CISC

Advantages of CISC architecture: (elprocus, n.d.)

- Each machine language instruction is grouped into a microcode instruction and executed accordingly, and then are stored inbuilt in the memory of the main processor, termed as microcode implementation.
- As the microcode memory is faster than the main memory, the microcode instruction set can be implemented without considerable speed reduction over hard-wired implementation.
- Entire new instruction set can be handled by modifying the microprogram design.
- CISC, the number of instructions required to implement a program can be reduced by building rich instruction sets and can also be made to use slow main memory more efficiently.
- Because of the superset of instructions that consists of all earlier instructions, this makes micro coding easy.

Drawbacks of CISC: (elprocus, n.d.)

- The amount of clock time taken by different instructions will be different – due to this – the performance of the machine slows down.
- The instruction set complexity and the chip hardware increases as every new version of the processor consists of a subset of earlier generations.
- Only 20% of the existing instructions are used in a typical programming event, even though there are many specialized instructions in existence which are not even used frequently.
- The conditional codes are set by the CISC instructions as a side effect of each instruction which takes time for this setting – and, as the subsequent instruction changes the condition code bits – so, the compiler has to examine the condition code bits before this happens.

RISC and CISC Convergence

The CISC's disadvantages are obvious. Although the main intention of the CISC processor architecture is to complete a task by using less number of assembly lines, the more complex instruction set of CISC machines require more time for fetching microcode-based control units that interpret instructions from memory. With fixed-length instructions, RISC lends itself to pipelining and speculative execution. RISC

RISC	CISC
Multiple register sets	Single register set
Three operands per instruction	One or two register operands per instruction
Parameter passing through register windows	Parameter passing through memory
Single-cycle instructions	Multiple cycle instructions
Hardwired control	Microprogrammed control
Highly pipelined	Less pipelined
Simple instructions, few in number	many complex instructions
Fixed length instructions	Variable length instructions
Complexity in compiler	Complexity in microcode
Only LOAD/STORE instructions access memory	many instructions can access memory
Few addressing modes	many addressing modes

Table 3
RISC vs CISC (Null & Lobur, 2012)

needs more RAM, whereas CISC has an emphasis on smaller code size and uses less RAM overall than RISC. Their differences are compared in Table. 3.

Nowadays, the major application of CISC is on the desktop computer, and RISC is widely used on mobile devices (with ARM chips). Many microprocessors today hold a mix of RISC- and CISC-like attributes (For example, Pentium), either RISC system provide more extravagant instruction set than old CISC systems, or combining the two in one processor.

Conclusion

The structure of the Pentium microprocessor is discussed in this article. Started with the Pentium family history, I introduced the design and an overview of a typical Pentium processor, with basic structures discussed in the introduction.

Then, several topics about microprocessors are discussed in details, with emphasize or exclusively relevant to Pentium. In the computer architecture, six-layer architecture and von Neumann Model are discussed. The Pentium D has dual-core von Neumann architecture. Then the branch prediction is introduced, which is used for optimizing the performance of branch waiting for waste. Examples for both static and dynamic branch prediction are given, as well as Pentium's Branch Target Buffer. Next, The features of Pentium architecture are listed. the next section is exclusively for Pentium, for its Rapid Execution Module. The following section is for memory subsystem, and the six-level memory is introduced, as well as an L2 cache in Pentium. Following that, Pentium's exclusive Hyper-threading technology is studied. In the last section, the advantages and drawbacks of both RISC and CISC are introduced. In the current market, it is becoming increasingly difficult to distinguish RISC architecture from CISC architecture.

References

Alpert, D., & Avnon, D. (1993, September). Architecture of the Pentium Microprocessor. *IEEE Micro*, 11–21.

Bryant, R. E., & O'Hallaron, D. R. (2015). *Computer Systems: A Programmer's Perspective*. Pearson.

Byte. (n.d.). How Prediction Works [Computer software manual].

Dandamudi, S. (2003). *Fundamentals of Computer Organization and Design*. Springer.

Das, S. P. (n.d.). Pentium Architecture [Computer software manual].

elprocus. (n.d.). Understanding about RISC and CISC Architectures [Computer software manual]. Retrieved from `https://www.elprocus.com/what-is-risc-and-cisc-architecture-and-their-workings/`

GeeksforGeeks. (n.d.). Branch Prediction in Pentium [Computer software manual].

Hesseldahl, A. (2000, July). *Why Cool Chip Code Names Die* (Tech. Rep.). Forbes.com. Retrieved from `http://www.ece.iastate.edu/~morris/cs570/name.html`

Hinton, G., Sager, D., Upton, M., Boggs, D., Carmean, D., Kyker, A., & Roussel, P. (2001). The Microarchitecture of the Pentium 4 Processor. *Intel Technology Journal*.

Intel. (2007). Product Change Notification [Computer software manual]. Retrieved from `https://qdms.intel.com/dm/i.aspx/B5883E95-206A-481F-8A33-A81B6E7F14D7/PCN107779-00.pdf`

Karbosguide. (n.d.). The Intel Pentium 4 prossor [Computer software manual].

Kolinko, E., & Lee, J. (n.d.). Branch Prediction [Computer software manual].

Marr, D. T., Binns, F., Hill, D. L., Hinton, G., Koufaty, D. A., Miller, J. A., & Upton, M. (2002). Hyper-Threading Technology Architecture and Microarchitecture. *Intel Technology Journal*.

McFarland, G. (2006). *Microprocessor Design: A Practical Guide from Design Planning to Manufacture*. McGraw-Hill.

Mittal, S. (2016). A Survey of Techniques for Dynamic Branch Prediction. *CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE*.

Null, L., & Lobur, J. (2012). *The Essentials of Computer Organization and Architecture*. Jones & Bartlett Learning, LLC.

pctechguide. (2018). *Basic structure of a Pentium microprocessor* (Tech. Rep.). Author. Retrieved from `https://www.pctechguide.com/cpu-architecture/basic-structure-of-a-pentium-microprocessor`

Shrestha, & Ram, P. (2014). Computer Essentials. *Kathmandu: Asmita's Publication*.

Stevens, P., & Viglas, S. (2009). *Computer Systems & Software Engineering* (Tech. Rep.). University of Edinburgh.

Stokes, J. (2002). Introduction to Multithreading, Superthreading and Hyperthreading. *Ars Technica*.

Vinyard, R. (2000). *Prediction* (Tech. Rep.). NMSU. Retrieved from `https://www.cs.nmsu.edu/~rvinyard/itanium/predication.htm`