Overview of Modern Processor Architecture

Parallel Execution Capabilities

JinHui Lin

ShenZhen University

2025-06-02

Today, I will discuss the design of modern processor architecture, which have a primary goal of improving performance, that is, executing more instructions in the same amount of time.

What Affects CPU Performance?

Instruction Count to be executed

This depends on:

- Program objectives
- ISA(Instruction Set Architecture)
- Code quality
- Programming language used
- Compiler behavior
- etc.

— What Affects CPU Performance?

Optional Given time constraints, I will skip the introduction and proceed directly to the main content

This is not related to our topic today, as we are discussing CPU microarchitecture design, not program design

What Affects CPU Performance?

Instruction Count to be executed

This depends on:

- Program objectives
- ISA(Instruction Set Architecture)
- Code quality
- Programming language used
- Compiler behavior
- etc.

Not within the scope of today’s discussion

— What Affects CPU Performance?

Optional Given time constraints, I will skip the introduction and proceed directly to the main content

This is not related to our topic today, as we are discussing CPU microarchitecture design, not program design

What Affects CPU Performance?

Clock Frequency

This depends on:

- Front-end CPU design
- Back-end design
- Manufacturing process
- etc.

— What Affects CPU Performance?

What Affects CPU Performance?

Clock Frequency

This depends on:

- Front-end CPU design
- Back-end design
- Manufacturing process
- etc.

Not within the scope of today's discussion

— What Affects CPU Performance?

What Affects CPU Performance?

IPC (Instructions Per Cycle)

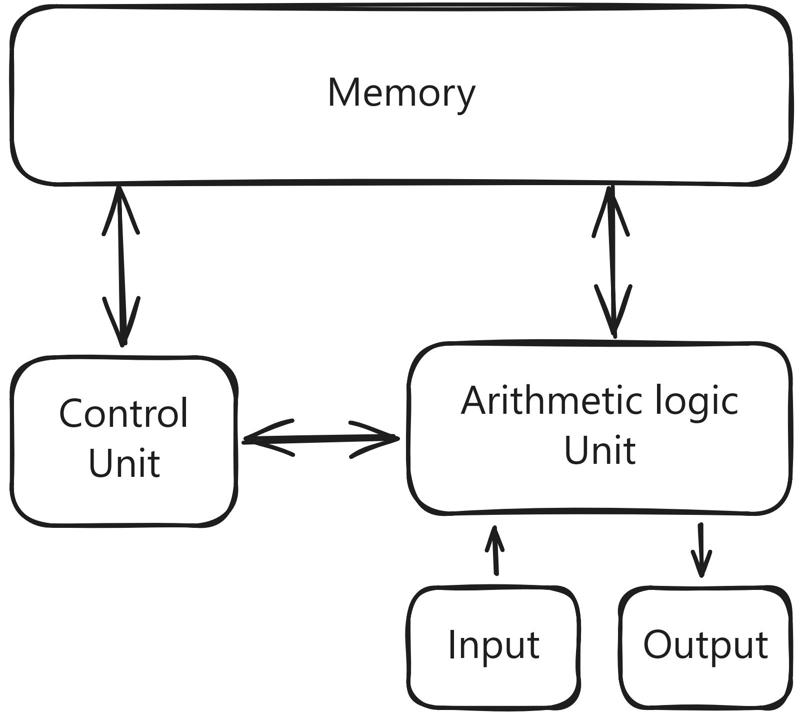
The number of instructions that can be executed per cycle

This essentially equates to—“the CPU’s ability to parallelize serial instructions”

— What Affects CPU Performance?

This essentially equates to—“the CPU’s ability to parallelize serial instructions”, which is the core of today’s discussion

Basic Computer Architecture



Memory

Control Unit

Arithmetic logic Unit

Input

Output

Classical Von Neumann Computer Architecture

— Basic Computer Architecture

This is the Von Neumann computer architecture,

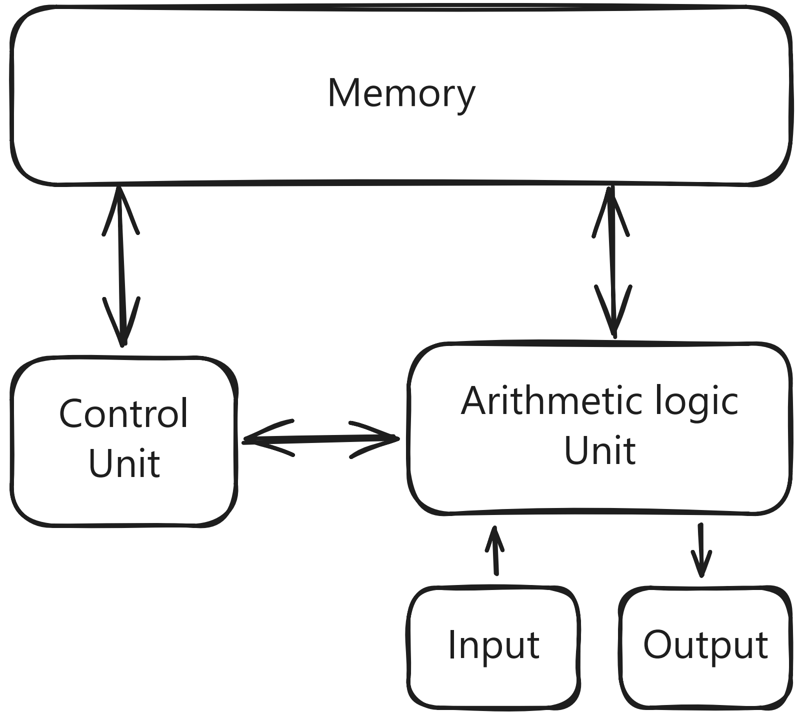
<pause>

which can still be used to explain computer behavior even today.

<pause>

Although there are minor differences in some areas, these are far beyond the scope of our discussion today. Today’s discussion only involves the internal design of the CPU.

Basic Computer Architecture

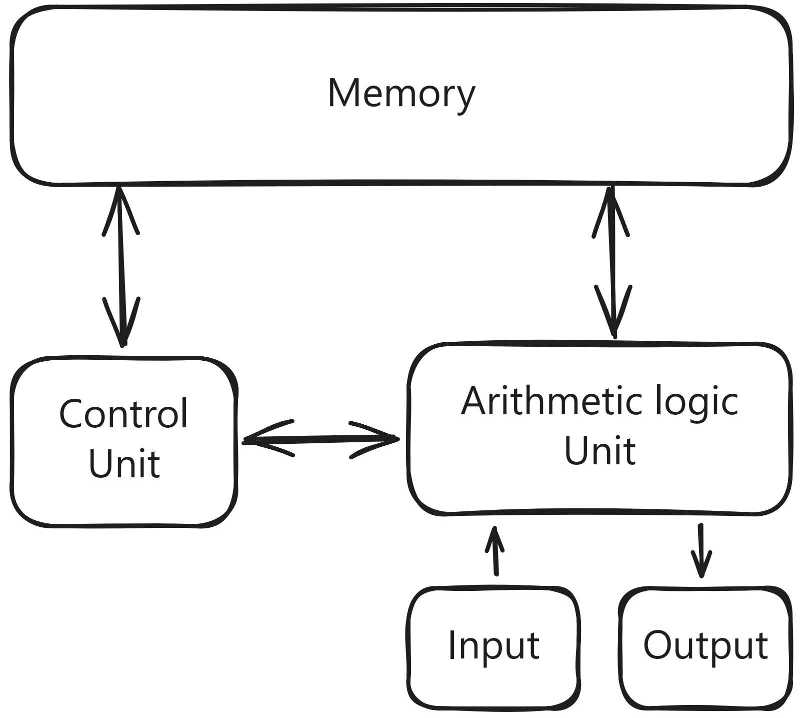


Classical Von Neumann Computer Architecture

Even today, this architecture can still be used to explain computer architecture.

— Basic Computer Architecture

Basic Computer Architecture



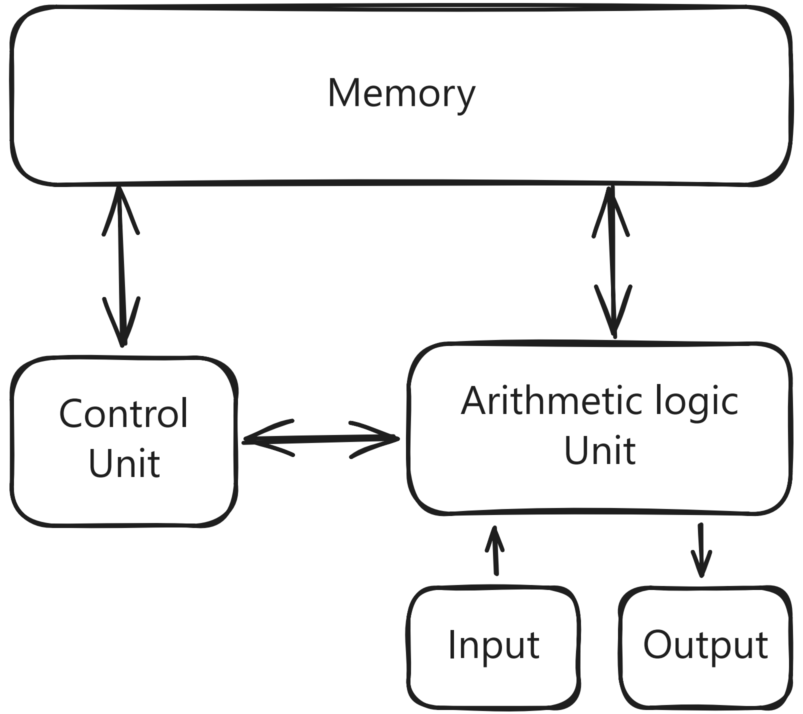
Classical Von Neumann Computer Architecture

Even today, this architecture can still be used to explain computer architecture.

Although there are minor differences in some areas (such as internal registers, cache, MMIO, etc.)

— Basic Computer Architecture

Basic Computer Architecture



Classical Von Neumann Computer Architecture

Even today, this architecture can still be used to explain computer architecture.

Although there are minor differences in some areas (such as internal registers, cache, MMIO, etc.)

Its operation process is essentially a cycle of “fetch->execute”

— Basic Computer Architecture

Pipelining

Pipelining is a classic CPU parallelization acceleration technique. By dividing CPU execution into multiple stages and executing these stages simultaneously, it increases frequency without significantly reducing IPC, thereby improving performance.

Modern CPUs rarely do not use pipelining technology. Even the microcontroller (MCU) in your washing machine likely has at least a two-stage pipeline (Arm CortexM0+).

— Pipelining

However, higher-performance CPUs often further subdivide the pipeline for higher frequency.

A classic pipeline stage division (you will likely see this in textbooks) is shown in the figure.

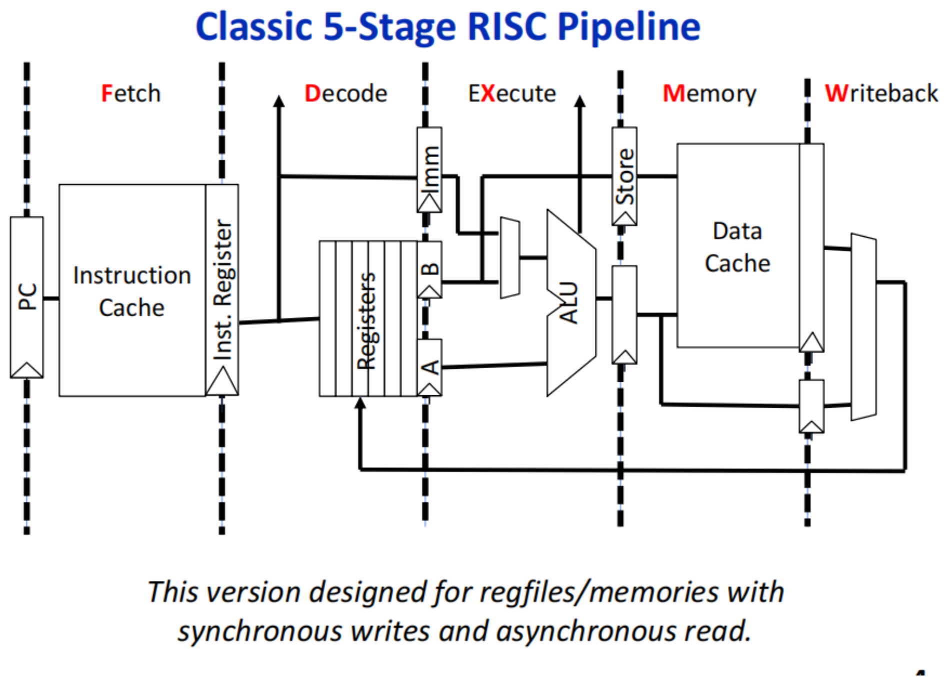
It basically divides instruction execution into five stages: “fetch,” “decode,” “execute,” “memory access,” and “writeback.”

<pause> However, pipelining cannot be executed arbitrarily.

Some special instruction and instrcution sequence may have data dependencies, which can lead to data hazards.

<pause>

Pipelining


Classic 5-Stage RISC Pipeline

This version designed for regfiles/memories with synchronous writes and asynchronous read.

A classic pipeline stage division

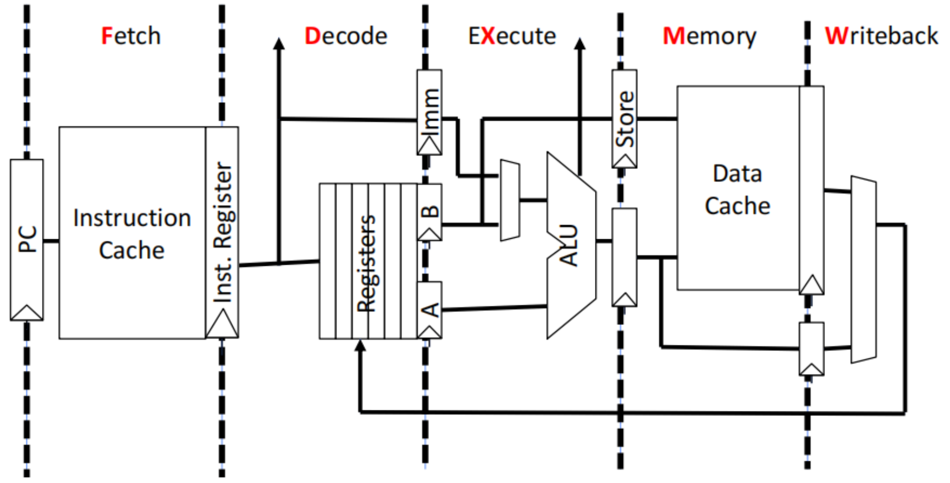
— Pipelining

Pipelining

— Pipelining

Pipelining

Classic 5-Stage RISC Pipeline



This version designed for regfiles/memories with synchronous writes and asynchronous read.

A classic pipeline stage division

However, pipelining cannot be executed arbitrarily.

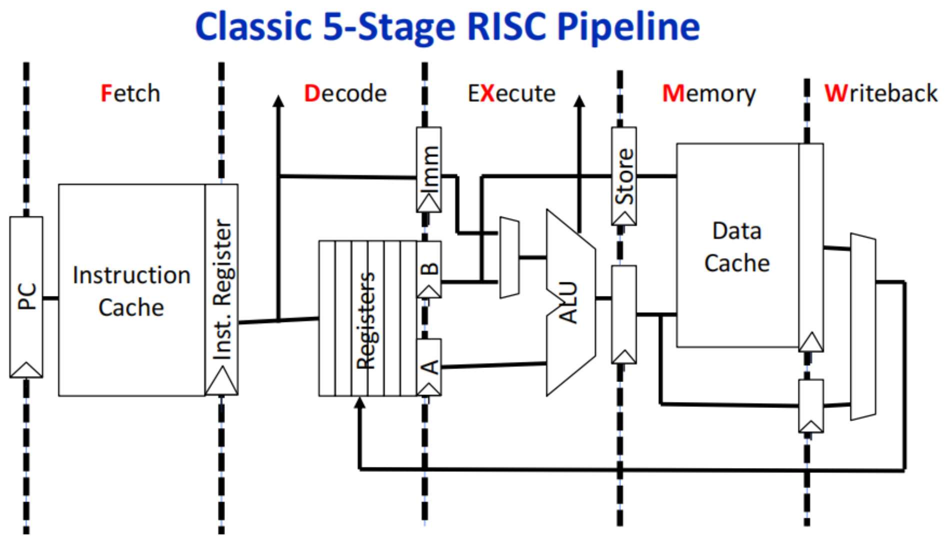
Some special instruction and instrcution sequence may have data dependencies, which can lead to data hazards.

— Pipelining

Pipelining

— Pipelining

Pipelining



This version designed for regfiles/memories with synchronous writes and asynchronous read.

A classic pipeline stage division

However, pipelining cannot be executed arbitrarily.

Some special instruction and instrcution sequence may have data dependencies, which can lead to data hazards.

The solution is through speculative execution, guessing the jump result, and flushing the pipeline if incorrect.

— Pipelining

Pipelining

Ok, this how pipeline works.

— Pipelining

Branch Prediction

If speculative execution is correct, it can significantly improve performance, so the accuracy of speculation becomes crucial.

— Branch Prediction

If speculative execution is correct, it can significantly improve performance, so the accuracy of speculation becomes crucial.

<pause>

The algorithm used to predict the branch jump direction is called branch prediction.

The Static branch prediction determines branch prediction direction at compile time.

<pause>

The Dynamic branch prediction records the historical jump results of branch instructions and uses these historical results to predict the next jump

Branch Prediction

If speculative execution is correct, it can significantly improve performance, so the accuracy of speculation becomes crucial.

Branch prediction algorithms:

- Static branch prediction
  Determines branch prediction direction at compile time.
- Dynamic branch prediction
  Records the historical jump results of branch instructions and uses these historical results to predict the next jump.

— Branch Prediction

If speculative execution is correct, it can significantly improve performance, so the accuracy of speculation becomes crucial.

<pause>

The algorithm used to predict the branch jump direction is called branch prediction.

The Static branch prediction determines branch prediction direction at compile time.

<pause>

The Dynamic branch prediction records the historical jump results of branch instructions and uses these historical results to predict the next jump.

Out-of-Order Execution

By reordering instructions, data hazards are avoided in advance.

— Out-of-Order Execution

Out-of-Order Execution

By reordering instructions, data hazards are avoided in advance.

Basic idea of out-of-order execution:

- Add an instruction queue
- Reorder it to avoid hazards
- After execution, write back in the original order

— Out-of-Order Execution

Superscalar

Superscalar refers to the CPU’s ability to execute multiple instructions in the same cycle.

— Superscalar

Remember the instruction queue mentioned above? We use it here because processors that implement superscalar often also implement out-of-order execution, as both require the processor to correctly determine dependencies between instructions.

Superscalar

Superscalar refers to the CPU’s ability to execute multiple instructions in the same cycle.

Basic idea of superscalar:

- Fetch multiple instructions simultaneously and push them into the instruction queue
- Based on instruction dependencies, push non-dependent instructions into the backend for simultaneous execution
- After execution, write back in the original order

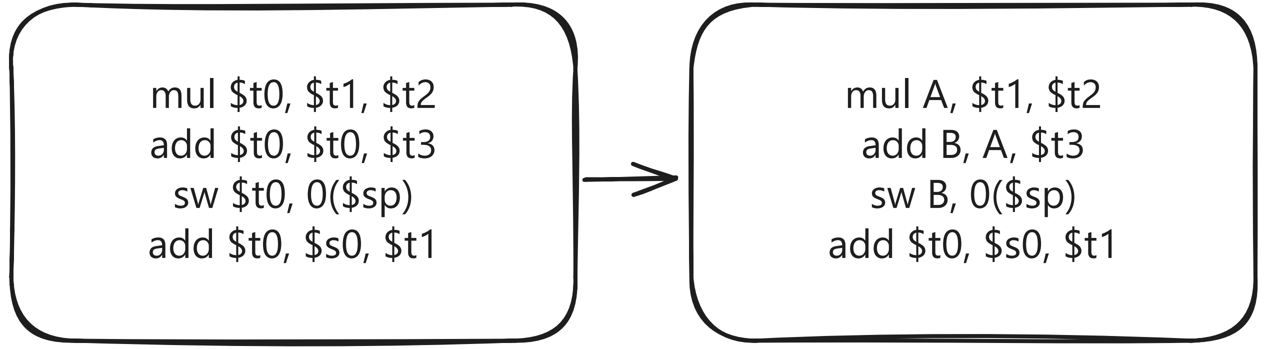
— Superscalar

Remember the instruction queue mentioned above? We use it here because processors that implement superscalar often also implement out-of-order execution, as both require the processor to correctly determine dependencies between instructions.

Register Renaming

Register renaming is another way to avoid hazards.

The basic idea is shown in the figure:



```
mul $t0, $t1, $t2
add $t0, $t0, $t3
sw $t0, 0($sp)
add $t0, $s0, $t1
```

```
mul A, $t1, $t2
add B, A, $t3
sw B, 0($sp)
add $t0, $s0, $t1
```

— Register Renaming

The basic idea of register renaming is to implement multiple physical registers for a single logical register, thereby avoiding WAR and WAW hazards. After solving these two hazards, the freedom of out-of-order execution is higher, allowing for higher instruction parallelism. RAW hazards can also be resolved through higher out-of-order execution.

Register Renaming

Tomasulo algorithm:

- Maintain a renaming table for each logical register
- When an instruction needs to write to a register, allocate a new physical register
- Update the renaming table, mapping the logical register to the new physical register
- Subsequent instructions reading this logical register will use the latest physical register
- When an instruction completes, release the physical register that is no longer needed

— Register Renaming

This renaming table is now often referred to as the ROB cache queue.

Summary

Key optimization technologies in modern processor architecture:

— Summary

Let’s summarize today’s discussion.

We mainly discussed several key technologies in modern processor architecture that improve performance.

These technologies all aim to solve the same problem: how to make the processor execute more instructions in a unit of time.

Summary

Key optimization technologies in modern processor architecture:

- Pipelining: Divides instruction execution into multiple stages for parallel execution
- Branch prediction: Reduces pipeline stalls by predicting branch jump direction
- Out-of-order execution: Reorders instructions to avoid data hazards
- Superscalar: Executes multiple instructions in the same cycle
- Register renaming: Avoids register hazards through physical register mapping

— Summary

Let’s summarize today’s discussion.

We mainly discussed several key technologies in modern processor architecture that improve performance.

These technologies all aim to solve the same problem: how to make the processor execute more instructions in a unit of time.

Thank You

If you have any questions, please feel free to discuss.

— Thank You

Thank you for listening.

If you have any questions, feel free to ask.