The Alpha 21264 microprocessor

François Costa - Seminar in computer architecture
R.E Kessler, E.J. McLellan, D.A. Webb
Compaq Computer Corporation, Shrewsbury, MA, USA

Executive summary

Goal : Design a processor that is able to run intensives applications like database, real-time visual computing, data mining, medical imaging

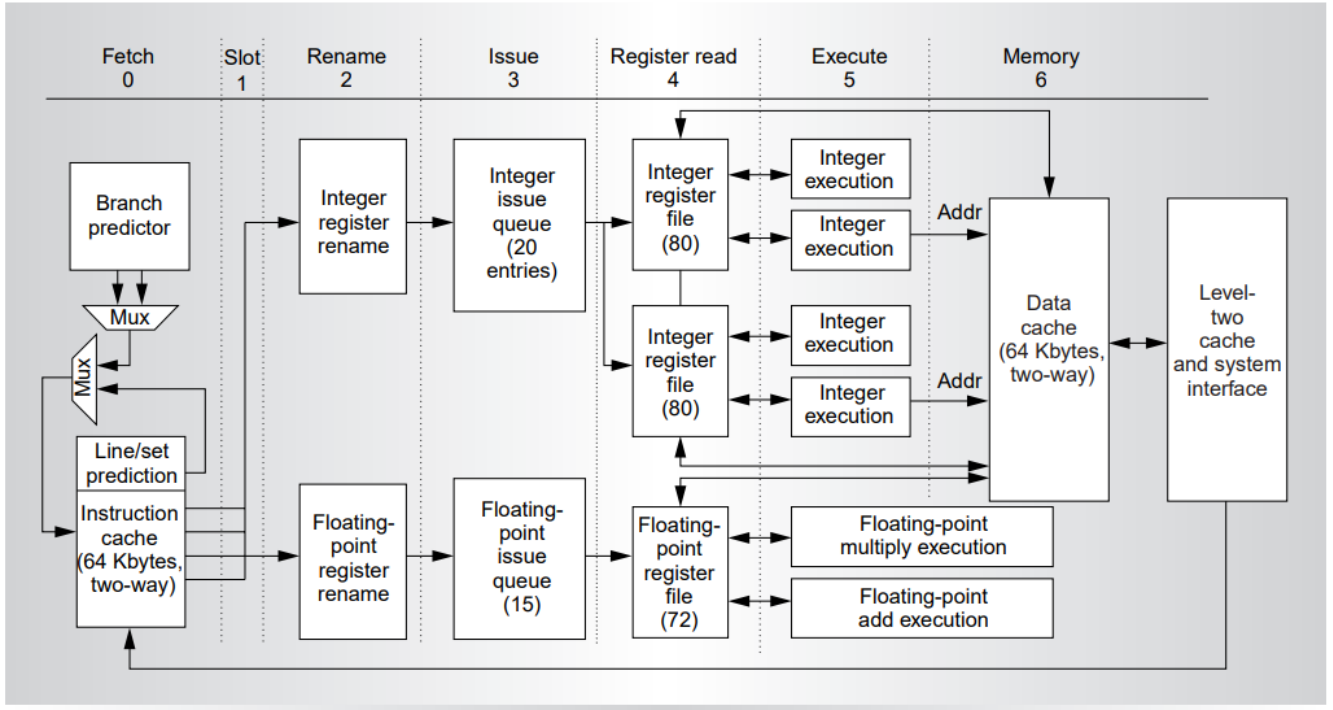
Problem :

1. Current processor execute instructions sequentialy. Hence limited parallelism
2. With pipelined design branches are problematic

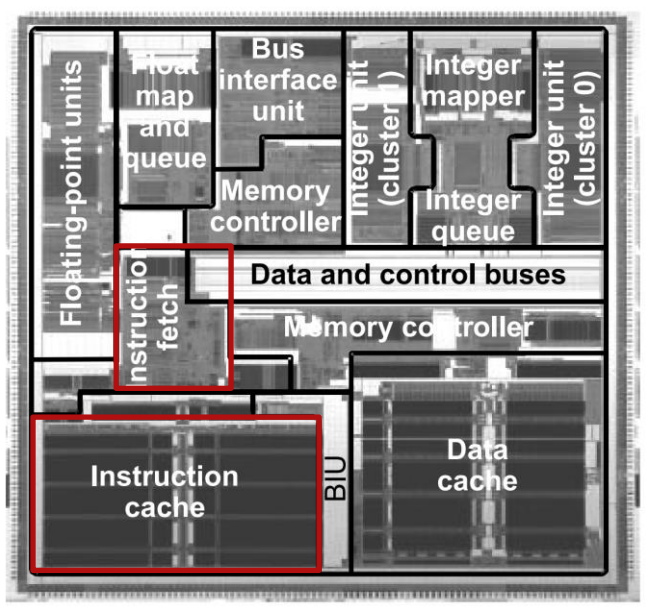
Idea :

1. Reorder instructions
2. Predict when data is not available

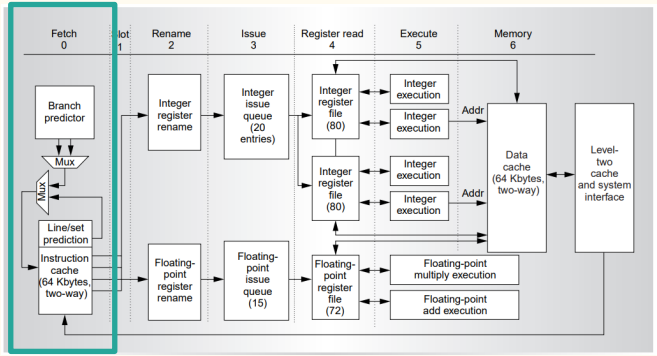
Overview of the Pipeline



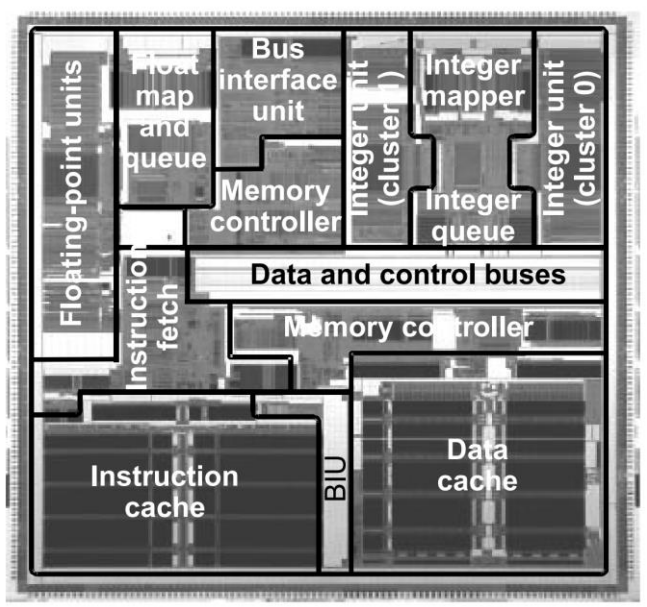
Overview of the Pipeline



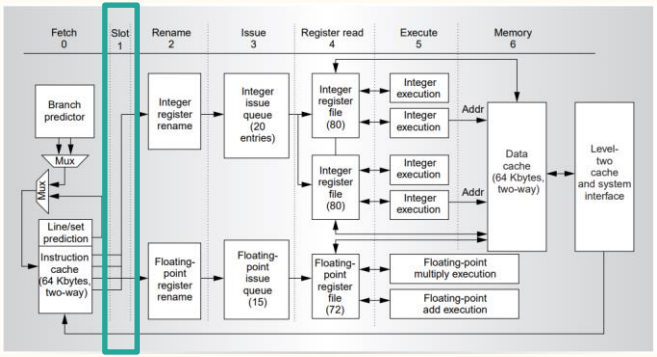
Stage 1
Fetch engine



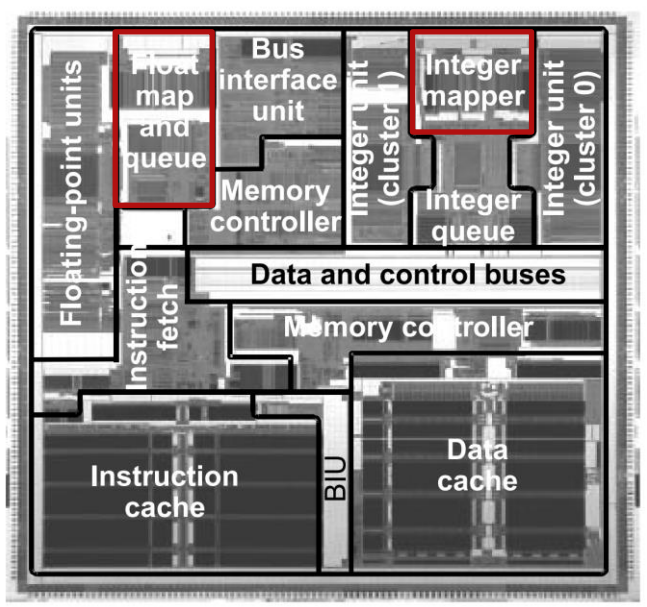
Overview of the Pipeline



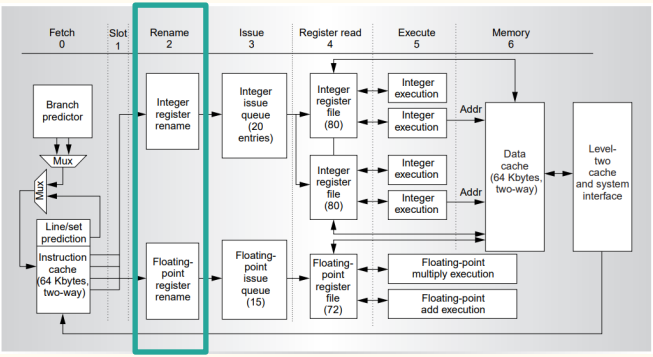
Stage 2
Split int and float operations



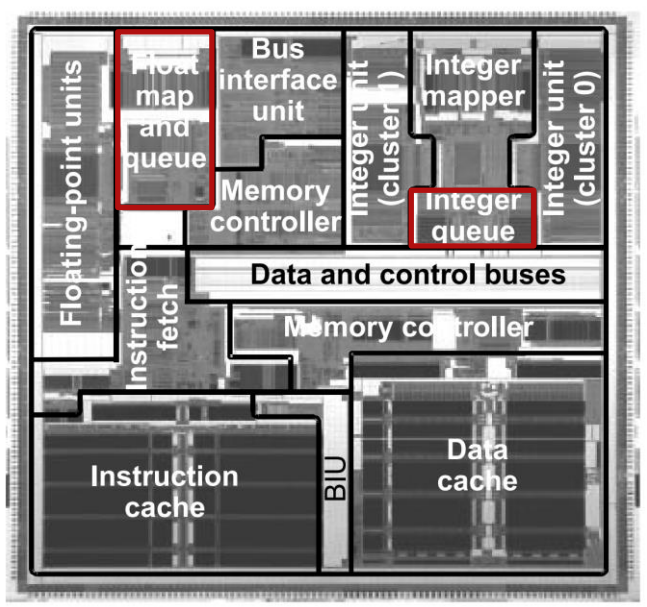
Overview of the Pipeline



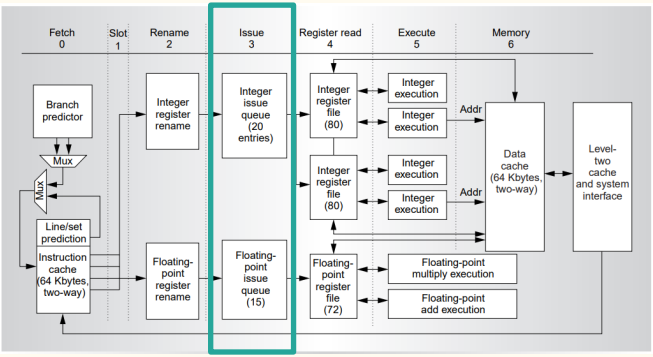
Stage 3
Rename engine



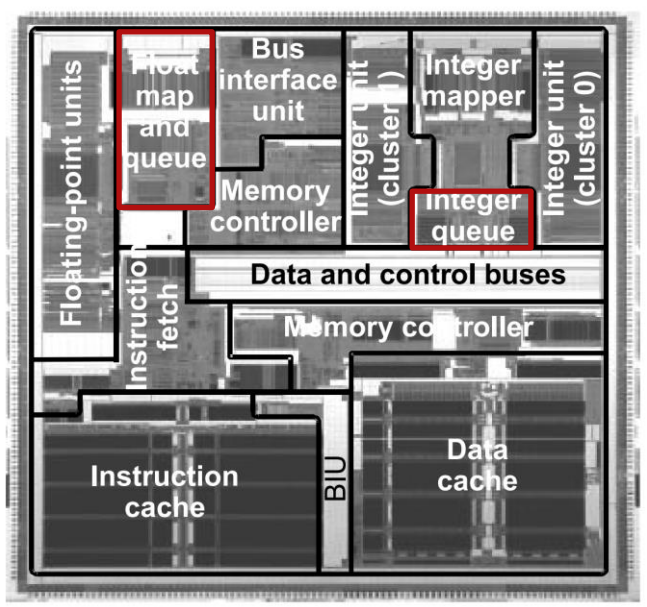
Overview of the Pipeline



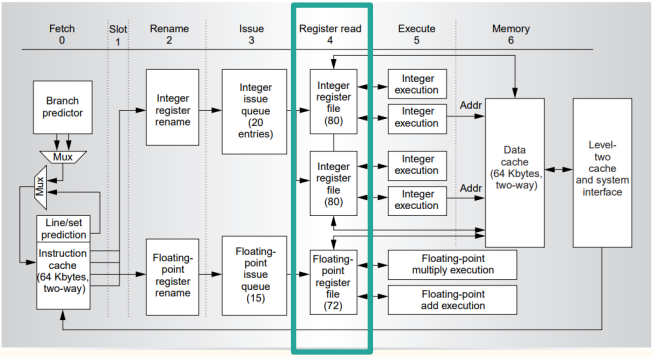
Stage 4
Issue queue



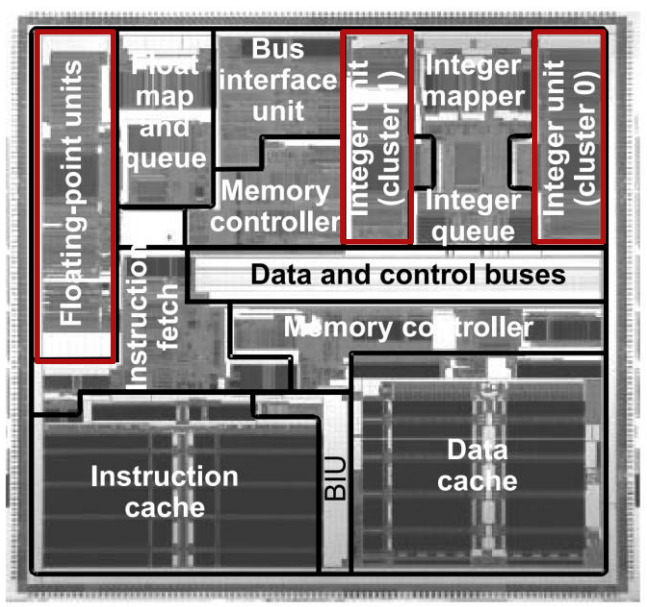
Overview of the Pipeline



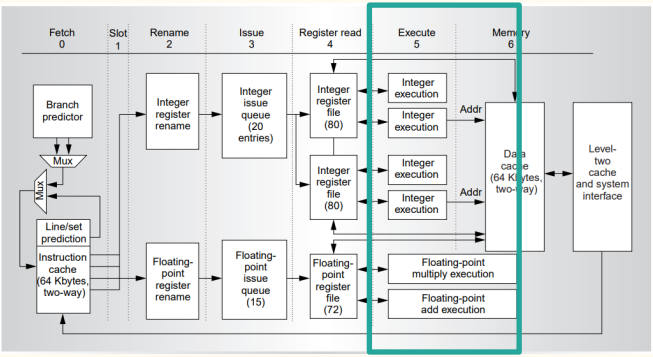
Stage 5
Register read



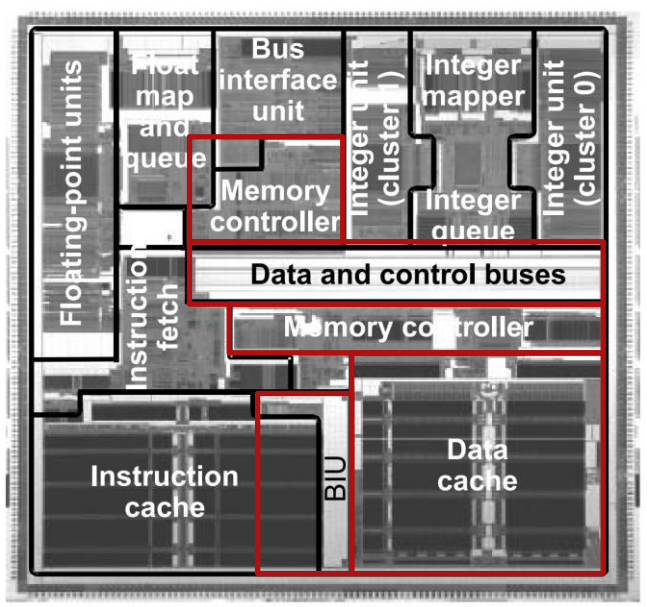
Overview of the Pipeline



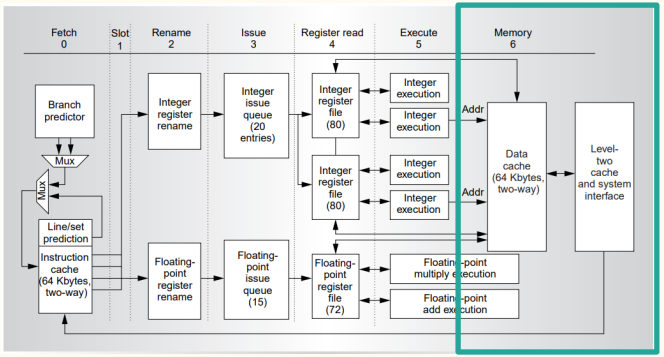
Stage 6
Execute



Overview of the Pipeline



Stage 7
Memory



Mechanism / Important features

Out of order execution

- Overview
- Register renaming
- Out of order issue queue
- Execution engine
- Instruction retire and exception handling

Speculative execution

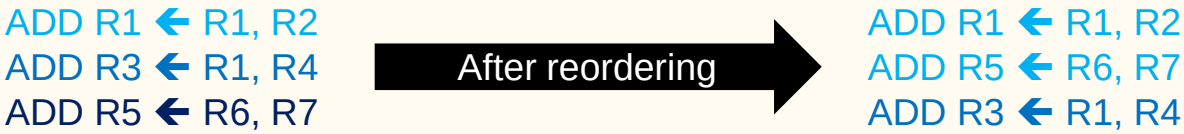
- Overview
- Branch prediction algorithm
- Line and way prediction
- Speculative loads
- Cache hit speculation

Other features

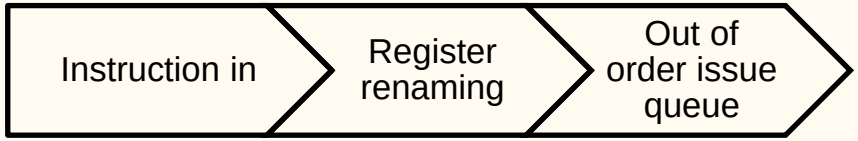
- Prefetching
- Internal memory system
- Bus interface unit

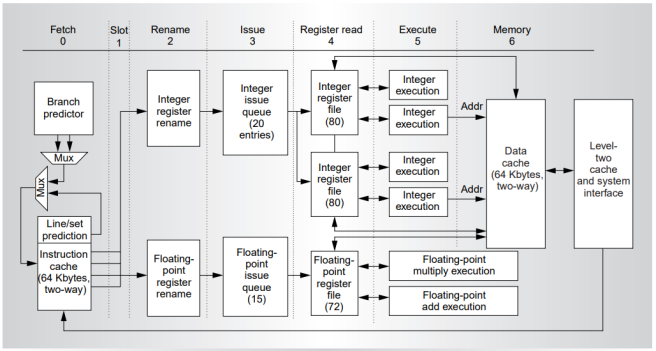
Out of order execution overview

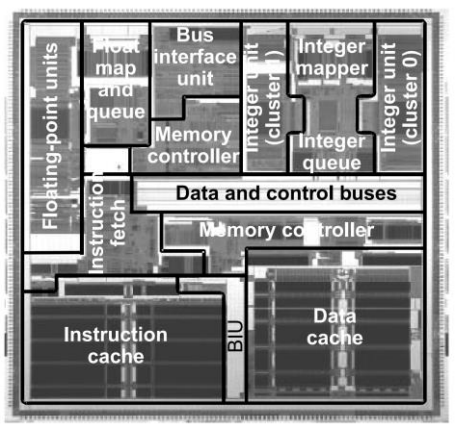
Out of order execution - Reorder instructions to increase parallelism

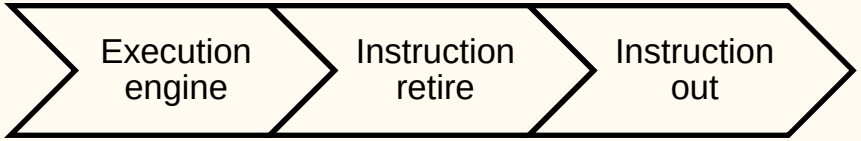


Mechanism - Out of order execution overview









Mechanism / Important features

Out of order execution

- Overview
- Register renaming
- Out of order issue queue
- Execution engine
- Instruction retire and exception handling

Speculative execution

- Overview
- Branch prediction algorithm
- Line and way prediction
- Fetch engine
- Speculative load
- Cache hit speculation

Other features

- Prefetching
- Internal memory system
- Bus interface unit

Mechanism - Register Renaming

- Eliminates write after write and write after read dependencies but keep read after write which is necessary
- 31 Float / Int visible registers and 41 Float/Int transparent registers
- Programmer only see when the instruction retires (invisible to the user)
- All next pipeline stages operate on the hidden register
- Register mapper store architectural state to restore in case of miss prediction

Mechanism / Important features

Out of order execution

- Overview
- Register renaming
- Out of order issue queue
- Execution engine
- Instruction retire and exception handling

Speculative execution

- Overview
- Branch prediction algorithm
- Line and way prediction
- Fetch engine
- Speculative load
- Cache hit speculation

Other features

- Prefetching
- Internal memory system
- Bus interface unit

Mechanism - out of order issue queue

- Two lists of separate queue floating point and integer instructions
- Can issue 4 int instructions per cycle and 2 float instructions per cycle
- Select oldest ready operation in order to execute less speculative instructions
- A scoreboard maintains the status of the register
- Queue logic selects each cycle available instructions

Mechanism / Important features

Out of order execution

- Overview
- Register renaming
- Out of order issue queue
- Execution engine
- Instruction retire and exception handling

Speculative execution

- Overview
- Branch prediction algorithm
- Line and way prediction
- Fetch engine
- Speculative load
- Cache hit speculation

Other features

- Prefetching
- Internal memory system
- Bus interface unit

Mechanism - Execution engine

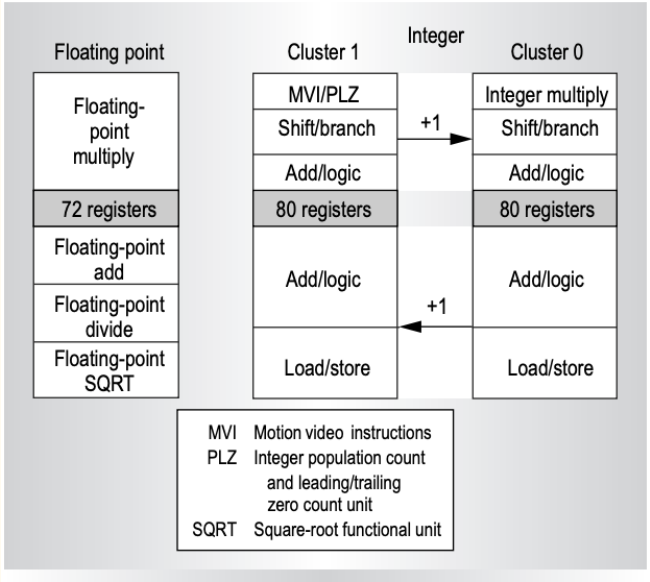
There are 6 execution engines

Use a clustered design

(-) Simpler but with a cycle overhead

(+) Critical path computation on the same cluster

(+) New motion video instructions (SIMD)



Mechanism / Important features

Out of order execution

- Overview
- Register renaming
- Out of order issue queue
- Execution engine
- Instruction retire and exception handling

Speculative execution

- Overview
- Branch prediction algorithm
- Line and way prediction
- Fetch engine
- Speculative load
- Cache hit speculation

Other features

- Prefetching
- Internal memory system
- Bus interface unit

Mechanism - Instruction retire and exception handling

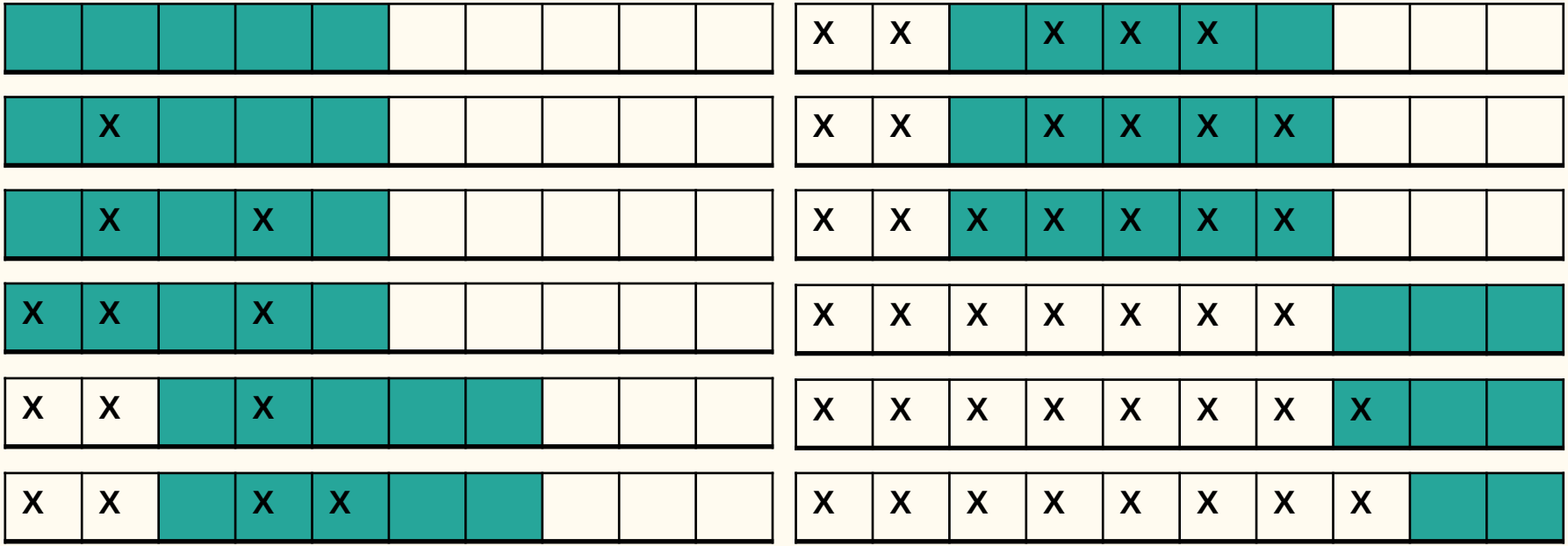
Mechanism :

1. Instructions are issued out of order but are retired in order
2. Each executed instruction are mapped in a in flight window (like for the TCP protocol)
3. The processor can retire when all instructions got executed before and no exception are generated ⇒ Non-speculative retirement

Characteristics :

1. 80 in-flight instructions + 32 in-flight load + 32 in-flight stores
2. Minimum latency Integer 4 Memory 7 Floating point 8 Branch / subroutine 7
3. Can retire up to 11 instructions per cycle

Sliding window protocol



Mechanism / Important features

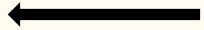
Out of order execution	Speculative execution	Other features
• Overview • Register renaming • Out of order issue queue • Execution engine • Instruction retire and exception handling	• Overview • Branch prediction algorithm • Line and way prediction • Fetch engine • Speculative load • Cache hit speculation	• Prefetching • Internal memory system • Bus interface unit

Speculative execution overview

```
string username = "Hello";
int password = 12345;

if(db[username] == password)
    do something
```


Execute i1


Execute i2


Execute i3


Should I execute i4 ?

One miss prediction costs 7 cycles which is a lot !

Mechanism / Important features

Out of order execution

- Overview
- Register renaming
- Out of order issue queue
- Execution engine
- Instruction retire and exception handling

Speculative execution

- Overview
- Branch prediction algorithm
- Line and way prediction
- Fetch engine
- Speculative load
- Cache hit speculation

Other features

- Prefetching
- Internal memory system
- Bus interface unit

Mechanism - Branch prediction algorithm

- Pattern style prediction
- Arbiter between local predictor and global predictor

Pattern style = Keep track for each branch history and assign taken or not taken for each branch history

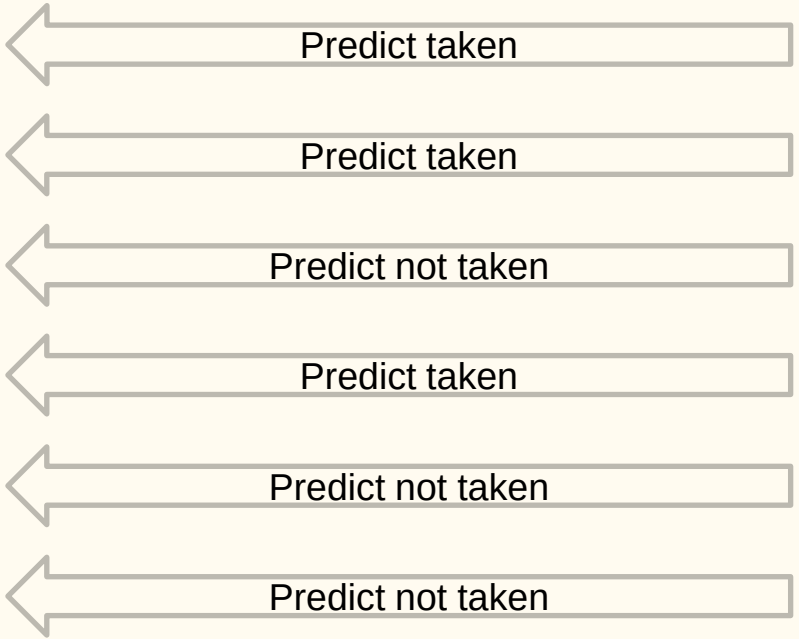
In this case, the local / global predictor is better

Local

Global

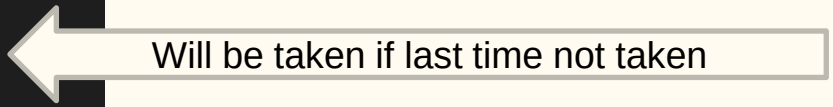
Pattern style branch predictor

T	T	T	T	T	T	T	T	T	T
T	T	N	T	T	N	T	N	T	N
N	T	N	T	N	T	N	T	N	T
T	N	T	N	T	N	T	N	T	N
N	N	N	N	N	T	T	T	T	T
N	N	N	N	N	N	N	N	N	N



Example where a local predictor is good

```
while(CONDITION) {
    i++;
    if(i % 2 == 0) {
        print("Even");
    }
}
```



Example where a global predictor is good

```
while(CONDITION) {
    i = random();
    if(i % 4 == 0) {
        print("You win 4$");
    }
    if(i % 3 == 0) {
        print("You win 3$");
    }
    if(i % 12 == 0) {
        print("You win 12$");
    }
}
```

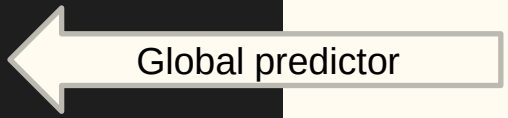
Assume this branch is taken

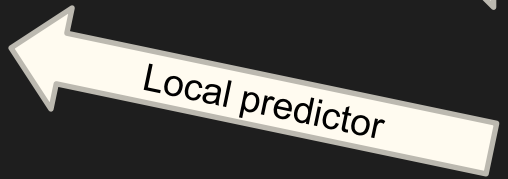
Assume this branch is taken

What about this branch ?

Example where an arbiter is important

```
int credits = 1000; // Player can play 1000 times
while(true) {
    i = random();
    if(i % 4 == 0)   print("You win 4$");
    if(i % 3 == 0)   print("You win 3$");
    if(i % 12 == 0)  print("You win 12$");
    if(credits == 0) break;
    credits--;
}
```





Mechanism / Important features

Out of order execution

- Overview
- Register renaming
- Out of order issue queue
- Execution engine
- Instruction retire and exception handling

Speculative execution

- Overview
- Branch prediction algorithm
- Line and way prediction
- Fetch engine
- Speculative load
- Cache hit speculation

Other features

- Prefetching
- Internal memory system
- Bus interface unit

Mechanism - Line and way prediction

The Alpha 21264 has a different cache architecture than the previous version of the processor (Alpha 21164)
The cache is a 64KB two-way set associative cache instead of 8 KB direct mapped instruction cache

Consequence : Better hit rates but with some bottleneck

Idea :Predict the next line of the cache and check in parallel

Most miss predictions costs 1 cycle but accuracy is between 85% and 100%

Mechanism / Important features

Out of order execution

- Overview
- Register renaming
- Out of order issue queue
- Execution engine
- Instruction retire and exception handling

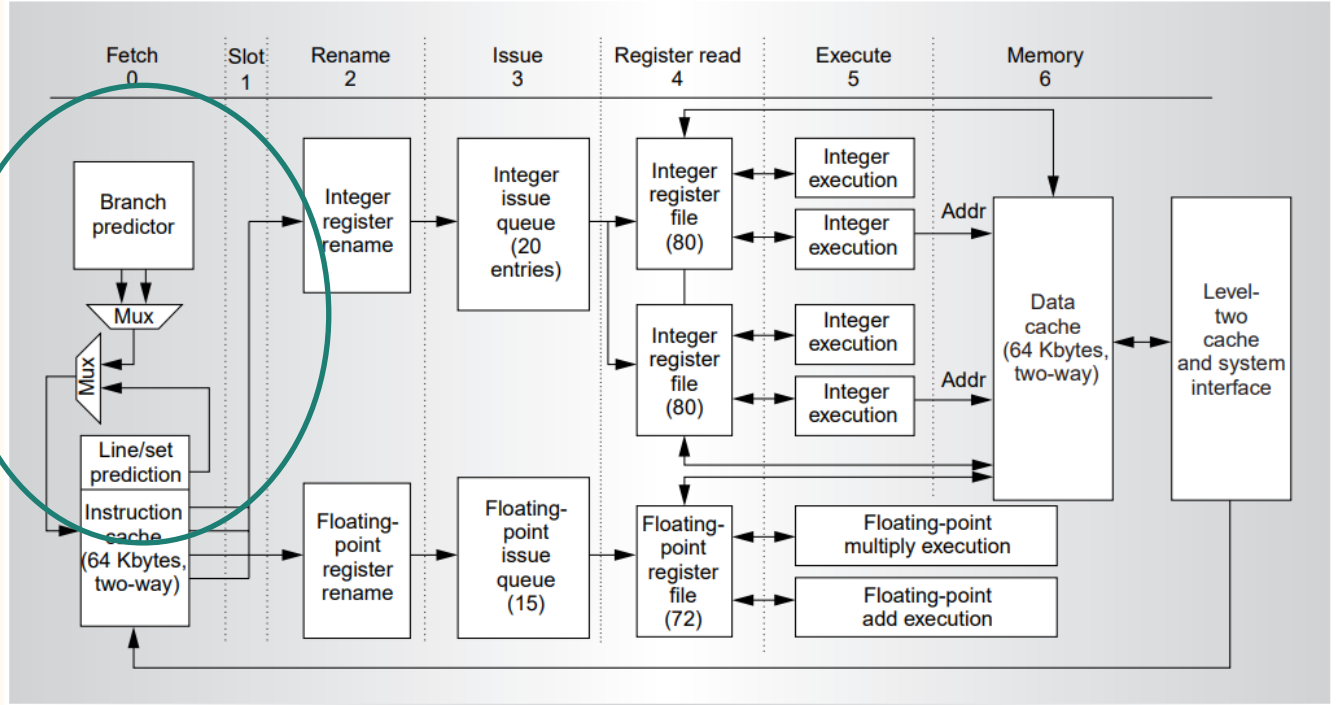
Speculative execution

- Overview
- Branch prediction algorithm
- Line and way prediction
- Fetch engine
- Speculative load
- Cache hit speculation

Other features

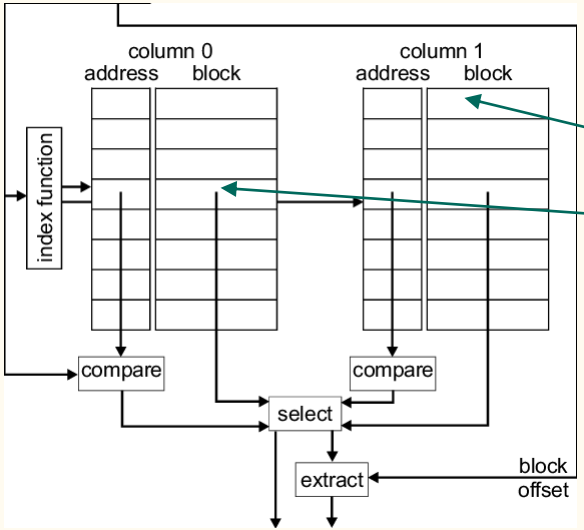
- Prefetching
- Internal memory system
- Bus interface unit

Fetch engine overview



Fetch engine overview

Fetch 4 instructions each cycle



Processor fetch using speculative mechanism

1. Branch prediction (what is the next instruction to fetch)
2. Line prediction (Which line contains the next instruction)

Which one should we choose ?

Mechanism / Important features

Out of order execution	Speculative execution	Other features
• Overview • Register renaming • Out of order issue queue • Execution engine • Instruction retire and exception handling	• Overview • Branch prediction algorithm • Line and way prediction • Speculative load • Cache hit speculation	• Prefetching • Internal memory system • Bus interface unit

Mechanism - Address and control structure

- Load come in the engine in order and instructions retire in order
- **Loads and stores can be reordered**
- Problem : An older store after a younger load can appear ➔ need a squash mechanism (more details in next slide)
- Speculative bypass of older store into younger load
- Eight entry miss address file tracks and forward miss to the bus interface unit

Mechanism - load store

Goal : Load as early as possible

Problem : Read after Write dependencies are hard to handle 

Solution : Train the mechanism to know which store can predicted or reordered

```
int arr[100];
arr[random_0_99] = 0x000CAFFE; // Almost impossible to predict
int variable = arr[1];
```

Mechanism / Important features

Out of order execution

- Overview
- Register renaming
- Out of order issue queue
- Execution engine
- Instruction retire and exception handling

Speculative execution

- Overview
- Branch prediction algorithm
- Line and way prediction
- Fetch engine
- Speculative load
- Cache hit speculation

Other features

- Prefetching
- Internal memory system
- Bus interface unit

Mechanism – speculative cache hit mechanism

- Speculation mechanism ⇒ three cycle load integer latency in the best case
- Assume that the data is already in the cache although it's not always case

  Result 1 : Correct ⇒ continue work normal

  Result 2 : Do a "mini restart" ⇒ two cycles latency penalty

The processor chooses to predict or not to predict depending on the application

Mechanism / Important features

Out of order execution

- Overview
- Register renaming
- Out of order issue queue
- Execution engine
- Instruction retire and exception handling

Speculative execution

- Overview
- Branch prediction algorithm
- Line and way prediction
- Fetch engine
- Speculative load
- Cache hit speculation

Other features

- Prefetching
- Internal memory system
- Bus interface unit

Mechanism - prefetching

Goal : Allow the programmer to take full benefit of the cache management and high bandwidth capacities. Very good for benchmarks with huge arrays.

Capacity : Prefetch 64-byte block to overlap cache miss time

How : Implemented via ISA instructions

Table 3. The 21264 cache prefetch and management instructions.

Instruction	Description
Normal prefetch	The 21264 fetches the 64-byte block into the L1 data and L2 cache.
Prefetch with modify intent	The same as the normal prefetch except that the block is loaded into the cache in a writeable state.
Prefetch and evict next	The same as the normal prefetch except that the block will be evicted from the L1 data cache on the next access to the same data cache set.
Write-hint 64	The 21264 obtains write access to the 64-byte block without reading the old contents of the block
Evict	The cache block is evicted from the caches.

Mechanism / Important features

Out of order execution

- Overview
- Register renaming
- Out of order issue queue
- Execution engine
- Instruction retire and exception handling

Speculative execution

- Overview
- Branch prediction algorithm
- Line and way prediction
- Fetch engine
- Speculative load
- Cache hit speculation

Other features

- Prefetching
- Internal memory system
- Bus interface unit

Mechanism - Internal memory system

- Execute up to two store / loads per cycle
- 32 in-flight load and store + 8 in-flight cache misses
- Use a two set associative 64 KB data cache which gives much better results than the previous one

⇒ Loads and stores can really exploit the out of order paradigm

Mechanism / Important features

Out of order execution

- Overview
- Register renaming
- Out of order issue queue
- Execution engine
- Instruction retire and exception handling

Speculative execution

- Overview
- Branch prediction algorithm
- Line and way prediction
- Fetch engine
- Speculative load
- Cache hit speculation

Other features

- Prefetching
- Internal memory system
- Bus interface unit

Mechanism - Bus interface unit

- It's the link between the internal memory system and the L2 cache and the rest of the system
- It takes MAF files as input and forward victim data (L3 cache) or data from main memory to the L2 or reverse
- Maintains cache coherent using invalidation cache coherence protocol
- Features : 12 cycles latency for L2 caches access and has a bandwidth of 1.3 GB / S

Key Results

“With more functional units and these dynamic execution techniques, the processor is 50% to 200% faster than its 21164 predecessor for many applications, even though both generations can fetch at most four instructions per cycle”

Summary

1. Use out of order execution to increase the amount of parallelism
2. Use a “modern” branch predictor
3. Use a lot of new predictions techniques like branch prediction, line prediction, prefetching, cache hit prediction
4. Use a more efficient cache
5. High throughput and low latency

Strengths

1. Clearly faster execution time
2. Can exploit implicit parallelism
3. Can run intensive workload like real-time visual computing app, database, etc..

“A unique combination of high clock speed and advanced microarchitectural techniques, including many forms of out-of-order and speculative execution, provide exceptional core computational performance in the 21264”

“Database, real-time visual computing, data mining, medical imaging, scientific/technical, and many other applications can utilize the outstanding performance available with the 21264.”

Weaknesses

1. Space & Cost overhead
2. Marketing paper
3. Could have bad performance on very special workloads due to the number of predictions
4. Use a lot of new techniques. They could have some issues that are unknown (spectre attack)

Takeaways

Out of order execution and Branch prediction can speedup a lot of applications. We clearly see that the performance of this processor is better thant the Alpha 21164 processor even if they both fetch 4 instructions per cycle

Indipendent to the programmer. Fully invisible at software level

High bandwidth and low latency data access

Open discussion

We have a lot of speculative execution. What can we do to improve the predicted rate ?

Open discussion

What could be done in the compiler in order to help the processor ?

Open discussion

What about more collaboration between software and hardware ?

Open discussion

Prefetching is implemented at ISA level. How should a programmer deal with it ?

Open discussion

Do you think that a programmer should care about instruction reordering or not ?