


20-Nov-06 (1)




CSC2510 - Computer Organization

Lecture 11: Floating Point and Pipelining

Philip Leong

20-Nov-06 (2)



IEEE Standard 754 Single Precision

Sign of number:
0 signifies +
1 signifies -

8-bit signed exponent in excess-127 representation

23-bit mantissa fraction


Value represented = $\pm 1.M \times 2^{E-127}$

(a) Single precision

Value represented = $1.001010 \dots \times 2^{-87}$

(b) Example of a single-precision number

20-Nov-06 (3)




IEEE 754 Numbers

- Normalized $\pm 1.d\dots d \times 2^{\text{exp}}$
- Denormalized $\pm 0.d\dots d \times 2^{\text{min_exp}}$

Format	# bits	#significant bits	macheps	#exponent bits	exponent range
Single	32	23+1	2^{-24} ($\sim 10^{-7}$)	8	$2^{-126} - 2^{127}$ ($\sim 10^{-38}$)
Double	64	52+1	2^{-53} ($\sim 10^{-16}$)	11	$2^{-1022} - 2^{1023}$ ($\sim 10^{-308}$)
Extended	≥ 80	≥ 64	$\leq 2^{-64}$ ($\sim 10^{-19}$)	≥ 15	$2^{-16382} - 2^{16383}$ ($\sim 10^{-4812}$)

Extended (80 bits on all Intel machines)
macheps = $2^{-\text{#significant bits}}$


20-Nov-06 (4)



Other Features

- $+, -, *, /, \text{sqrt}, \text{remainder}$, conversion to and from integer are correctly rounded
 - As if computed with infinite precision and then rounded
 - Transcendental functions are not correctly rounded (Table Maker's dilemma)
- Invalid Operation, Overflow, Division by zero, Underflow, Inexact result exceptions and flags
- FP numbers can be treated as integers for comparisons
- $+/-$ ($+0 == -0$)
 - E.g. $1/(1/x) = x$, complex
- Infinity** is like the mathematical one
 - finite / Infinity $\rightarrow 0$
 - Infinity * Infinity \rightarrow Infinity
 - Nonzero / 0 \rightarrow Infinity
 - Infinity^(finite or Infinity) \rightarrow Infinity
- NaN** is produced whenever a limiting value cannot be determined:
 - Infinity - Infinity \rightarrow NaN
 - Infinity / Infinity \rightarrow NaN
 - 0 / 0 \rightarrow NaN
 - Infinity * 0 \rightarrow NaN
 - Signalling and quiet (many systems only have quiet)
- Trivia question: when does $x != x$?


20-Nov-06 (5)



Special Values

- Exponents of 0 and 255 have special meaning
 - $E=0, M=0$ represents 0 (sign bit still used so there is $+/-0$)
 - $E=0, M <> 0$ is a denormalised number $\pm 0.M \times 2^{-126}$ (smaller than the smallest normalised number)
 - $E=255, M=0$ represents \pm infinity
 - $E=255, M <> 0$ represents NaN (not a number e.g. returned for $0/0$ or $\text{sqrt}(-1)$)

20-Nov-06 (6)



Example

- What is the single precision number 40C0,0000 in decimal?
- What is -0.5 in binary?

20-Nov-06 (7)



Example

- What is the single precision number 40C0,0000 in decimal?
 - 01000000110000000000000000000000
 - Exp=2, Mantissa = 1.5, Decimal 6
- What is -0.5 in binary?
 - Sign=-1, Exp=-1, Mantissa=1
 - 10111111000000000000000000000000
 - BF000000

20-Nov-06 (8)



Floating point not always correct

- Find 2nd root of
 - $r = (-b - \sqrt{b^2 - 4ac}) / (2a)$
- Sparc processor, Solaris, gcc 3.3 (ANSI C),
- ```
EXACT 0.00023025562642476431
DOUBLE 0.00023025562638524986
FLOAT 0.00024670246057212353
```
- Problem is that  $\sqrt{b^2 - 4ac} \approx -b$
  - Rule of thumb: use the highest precision which does not give up too much speed

20-Nov-06 (9)



## Catastrophic Cancellation

- (a-b) is inaccurate when  $a \approx b$
- Examples (decimal)
  - Using 2 significant digits compute mean of 5.1 and 5.2 using the formula  $(a+b)/2$ .  
 $a+b=10, 10/2=5$  (mean is less than both numbers)
  - Using 8 significant digits  
 $(11111113+11111111)+7.5111111=9.5111111$   
 $11111113+(-11111111)+7.5111111=10.000000$
- Catastrophic cancellation occurs when

$$\left| \frac{[\text{round}(x) \bullet \text{round}(y)] - \text{round}(x \bullet y)}{\text{round}(x \bullet y)} \right| \gg \epsilon_{mach}$$

20-Nov-06 (10)

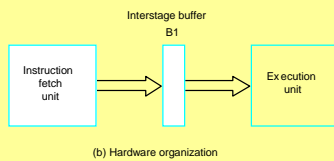
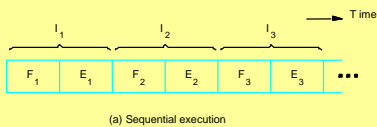


## Pipelining (not examinable)

20-Nov-06 (11)



## Sequential Execution



20-Nov-06 (12)



## Pipelining

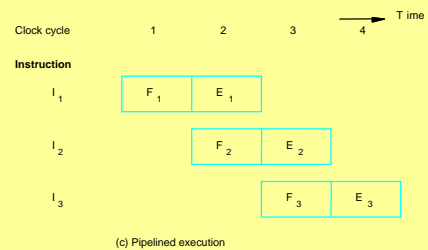


Figure 8.1. Basic idea of instruction pipelining.

20-Nov-06 (13)

## Pipelining

- Parallelism is increased by overlapping fetch and execute
- More is better (4 stage pipeline)
  - Fetch instruction from memory
  - Decode instruction and fetch source operands
  - Execute instruction
  - Write results

20-Nov-06 (14)

## 4 stage pipeline

- Each stage should be roughly of the same maximum period (why?)
- What is the overhead?
- What is the speedup of an N stage pipeline?

(a) Instruction execution divided into four steps

(b) Hardware organization

20-Nov-06 (15)

## Reality

- A pipeline stage may require more than 1 cycle (E2), others have to wait (pipeline **stalled**)

20-Nov-06 (16)

## Stalls

- Any condition that causes pipeline to stall called a **hazard** e.g. cache miss in F2

| Clock cycle | 1 | 2              | 3              | 4              | 5    | 6              | 7              | 8              | 9              |
|-------------|---|----------------|----------------|----------------|------|----------------|----------------|----------------|----------------|
| Stage       | F | F              | F              | F              | F    | F              | F              | F              | F              |
| D: Decode   |   | D <sub>1</sub> | idle           | idle           | idle | D <sub>2</sub> | D <sub>3</sub> |                |                |
| E: Execute  |   |                | E <sub>1</sub> | idle           | idle | idle           | E <sub>2</sub> | E <sub>3</sub> |                |
| W: Write    |   |                |                | W <sub>1</sub> | idle | idle           | idle           | W <sub>2</sub> | W <sub>3</sub> |

(b) Function performed by each processor stage in successive clock cycles

20-Nov-06 (17)

## Data Hazards

- Consider  $A=3+A$ ;  $B=4*A$  (delayed by 2 cycles)

Figure 8.6. Pipeline stalled by data dependency between D<sub>2</sub> and W<sub>1</sub>.

20-Nov-06 (18)

## Software solution to data hazard

```

I1: Mul R2,R3,R4
 NOP
 NOP
I2: Add R5,R4,R6

```

Advantage: hardware simple  
Disadvantage: slow

20-Nov-06 (19)

## Forwarding as a solution

Figure 8.7. Operand forwarding in a pipelined processor

20-Nov-06 (20)

## Branches

- Approx 20% of instructions so must be efficient

(a) Branch address computed in Execute stage (b) Branch address computed in Decode stage

20-Nov-06 (21)

## Branch detected in decode stage

(a) Branch address computed in Execute stage (b) Branch address computed in Decode stage

Figure 8.9. Branch timing.

20-Nov-06 (22)

## Instruction Queue

- Fetch unit can recognize and execute branches
- Fetch gets one instruction, dispatch consumes one instruction

20-Nov-06 (23)

## Branch timing

Figure 8.11. Branch timing in the presence of an instruction queue. Branch target address is computed in the D stage.

20-Nov-06 (24)

## Branch Timing

- I1 takes 3 E cycles, I6 discarded
- D4 can execute in its place (saves 1 cycle)
- Branch computed in parallel with other instructions so no cycles lost due to branch (called **branch folding**)
- Instruction queue can also hide effect of cache miss

## Conditional Branches

- Introduce an additional problem due to dependency of branch on the result
- Delayed branching** can minimize this penalty

|      |            |       |
|------|------------|-------|
| LOOP | Shift_left | R1    |
|      | Decrement  | R2    |
|      | Branch=0   | LOOP  |
| NEXT | Add        | R1,R3 |

(a) Original program loop

|      |            |       |
|------|------------|-------|
| LOOP | Decrement  | R2    |
|      | Branch=0   | LOOP  |
|      | Shift_left | R1    |
| NEXT | Add        | R1,R3 |

(b) Reordered instructions

## Delayed Branch

## Branch Prediction

- Attempt to predict conditional branch direction
- Use **speculative execution** so if we get it right, no lost cycles
- Do not update registers/memory until we know we got it right, if we get it wrong, cancel the instructions
- Branch prediction can be dynamic or static

## Superscalar Operation

- Execute multiple instructions at any time via multiple processing units (can execute > 1 instruction/cycle)

## What slows down a processor?

$r1 \leftarrow \text{mem}[r0] /* \text{Instruction 1} */$   
 $r2 \leftarrow r1 + r2 /* \text{Instruction 2} */$   
 $r5 \leftarrow r5 + 1 /* \text{Instruction 3} */$   
 $r6 \leftarrow r6 - r3 /* \text{Instruction 4} */$

- Cache miss stalls entire processor for 20-30 cycles
- Instruction 2 cannot execute because it needs r1
- Can look ahead and find instructions to execute (instruction 3 and 4)
- Instructions executed out of order
- IA32 doesn't have many registers, many false dependencies will occur and cause stalling
- Registers mapped to larger register set (renamed) to allow more forward progress
- Retire unit can write to the real register file and these are done in original program order

## Dynamic execution

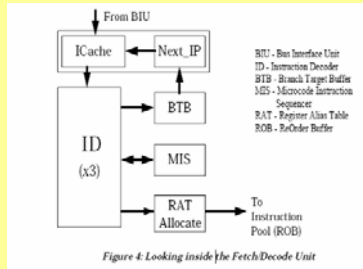
- Predict program flow (including conditional branches)
- Analyze data dependencies and speculatively execute instructions in best order

20-Nov-06 (31)



## Pentium P6

- FETCH/DECODE unit:** An in-order unit that takes as input the user program instruction stream from the instruction cache, and decodes them into a series of micro-operations (uops) that represent the dataflow of that instruction stream. The program pre-fetch is itself speculative.



Ref: [A 0.6um BiCMOS Processor with Dynamic Execution](#) ISSCC 95; pp 176-177

20-Nov-06 (32)



## Pentium P6

- The 512 entry Branch Target Buffer (BTB) helps the Instruction Fetch Unit (IFU) choose an instruction cache line for the next instruction fetch. ICache line fetches are pipelined with a new instruction line fetch commencing on every CPU clock cycle.
- Three parallel decoders (ID) convert multiple Intel Architecture instructions into multiple sets of micro-ops (uops) each clock.
- The sources and destinations of these uops are renamed by the Register Alias Table (RAT), which eliminates register re-use artifacts, and are forwarded to the Reservation Station (RS) and to the ReOrder Buffer (ROB).

20-Nov-06 (33)



## Pentium P6

- DISPATCH/EXECUTE unit:** An out-of-order unit that accepts the dataflow stream, schedules execution of the uops subject to data dependencies and resource availability and temporarily stores the results of these speculative executions.

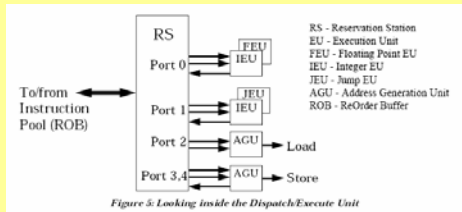


Figure 5: Looking inside the Dispatch/Execute Unit

20-Nov-06 (34)



## Pentium P6

- The renamed uops are queued in the RS where they wait for their source data - this can come from several places, including immediates, data bypassed from just-executed uops, data present in a ROB entry, and data residing in architectural registers (such as EAX).
- The queued uops are dynamically executed according to their true data dependencies and execution unit availability (IEU, FEU, AGU). The order in which any given uops execute in time has no particular relationship to the order implied by the source program.
- Memory operations are dispatched from the RS to the Address Generation Unit (AGU) and to the Memory Ordering Buffer (MOB). The MOB ensures that the proper memory access ordering rules are observed.

20-Nov-06 (35)



## Pentium P6

- RETIRE unit:** An in-order unit that knows how and when to commit ("retire") the temporary, speculative results to permanent architectural state.
- BUS INTERFACE unit:** A partially ordered unit responsible for connecting the three internal units to the real world. The bus interface unit communicates directly with the L2 cache supporting up to four concurrent cache accesses. The bus interface unit also controls a transaction bus, with MESI snooping protocol, to system memory.

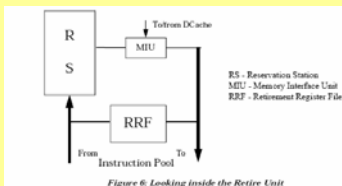


Figure 6: Looking inside the Retire Unit

20-Nov-06 (36)



## Pentium P6

- Once a uop has executed, and its destination data has been produced, that result data is forwarded to subsequent uops that need it, and the uop becomes a candidate for "retirement".
- Retirement hardware in the ROB uses uop timestamps to reimpose the original program order on the uops as their results are committed to permanent architectural machine state in the Retirement Register File (RRF). This retirement process must observe not only the original program order, it must correctly handle interrupts and faults, and flush all or part of its state on detection of a mispredicted branch. When a uop is retired, the ROB writes that uop's result into the appropriate RRF entry and notifies the RAT of that retirement so that subsequent register renaming can be activated.

20-Nov-06 (37)



## Intel Core Duo

- (10 years after P6)
- Recent trend is to use all of the techniques described today plus multiple cores on a chip

