Electrical & Computer ENGINEERING

Architecture Impact on Performance

18-645: How to write fast code

Tze Meng Low

1

Electrical & Computer ENGINEERING

SOFTWARE

18645 – How to write fast code

HARDWARE

2

What is software…

C/C++

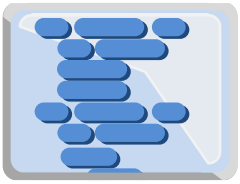
```
for (int i = 0; i != m; ++i)
  for (int j = 0; j!= n; ++j)
    for (int k = 0; k != p; ++k)
      c[j*m+i] += a[p*m+i] * b[j*p+k];
```

Python

```
C = A @ B
```


3

Software wrote is not software executed



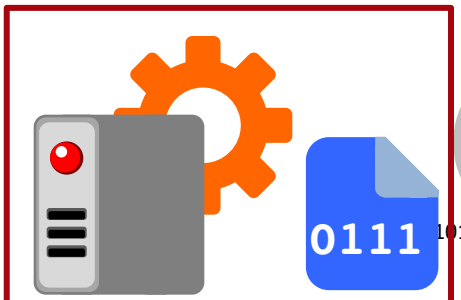
```
C = A @ B
```

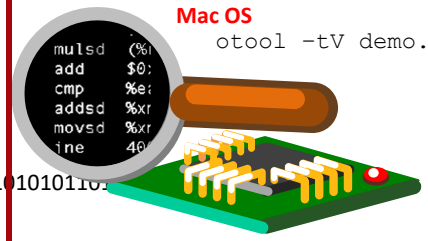
Linux

```
objdump -d demo.x
```

Mac OS

```
otool -tV demo.x
```





Compilation/Interpretation

4

Instruction Set Architecture

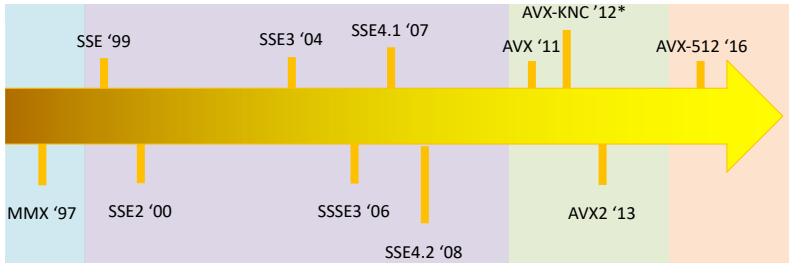
- Interface between hardware and software
  - Lowest level of control for the programmer
  - Exposes program details such as
    - Program state
    - Instructions used
    - Registers
  - Relatively stable for many years / decades

5

Instruction Set Architecture

- ISA can change to introduce new capabilities

x86 SIMD Instruction Set Over The Years

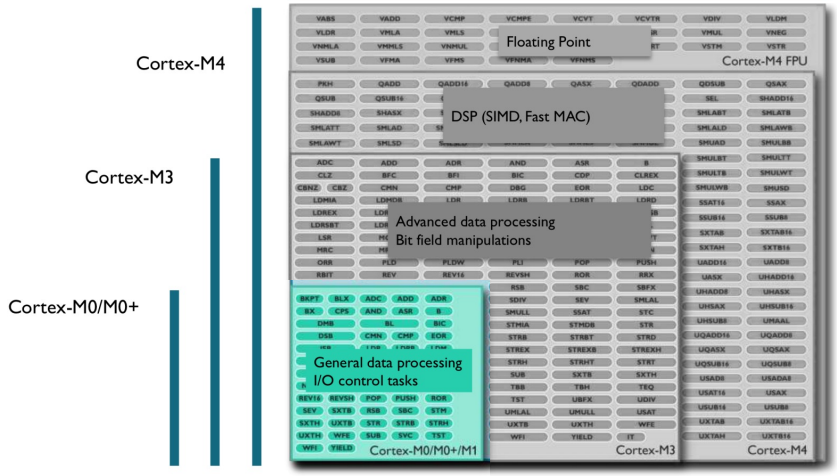


SIMD Family	MMX	SSE	AVX	AVX-512
Data Length (bit)	64	128	256	512
Data Length (floats)	2	4	8	16

6

Instruction Set Architecture

ARM Cortex-M Instruction Set Architecture

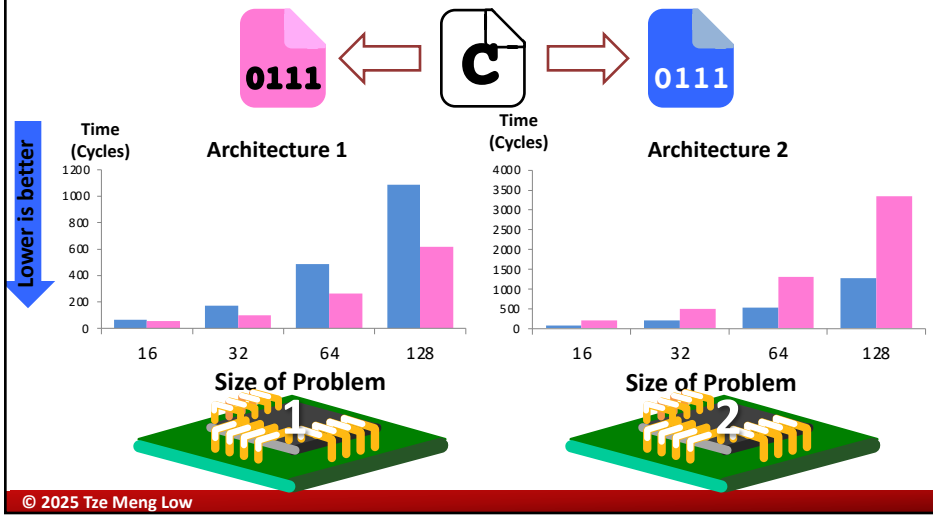


http://www.anandtech.com/show/8400/arms-cortex-m-even-smaller-and-lower-power-cpu-cores

7

Programming at the ISA may not be fast

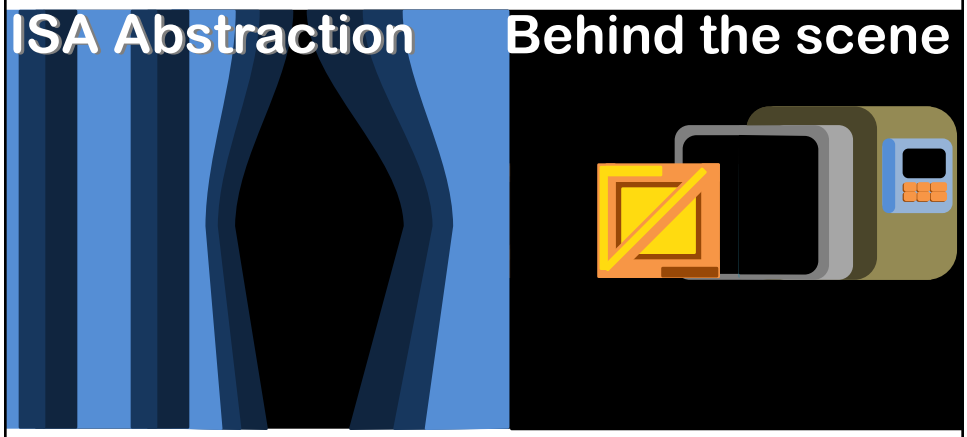
- Different performance due to HW differences



8

ISA is an abstraction

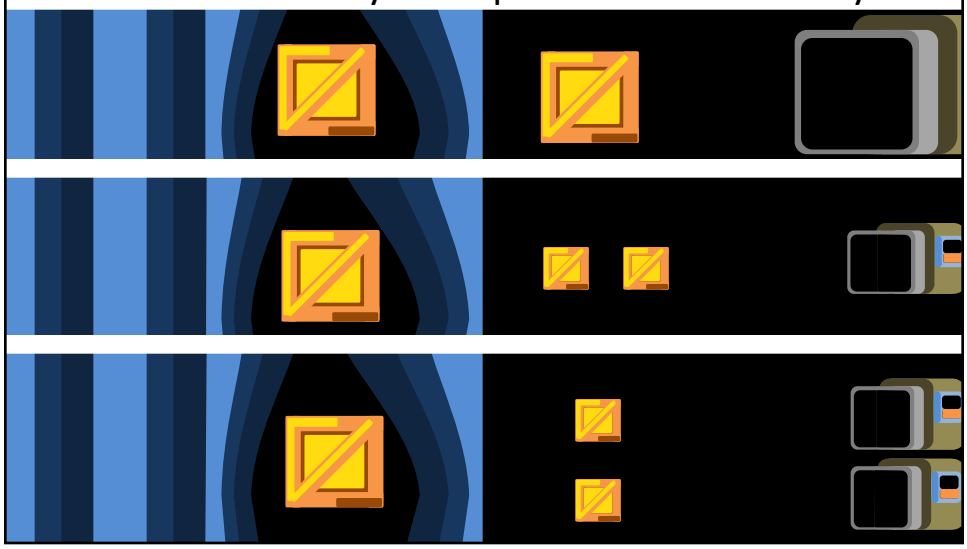
- Instructions may be executed in different order



9

ISA is an abstraction

- Instructions may be implemented differently



10

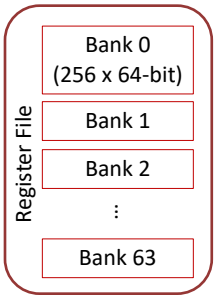
ISA is an abstraction

- Logical registers vs physical registers

Small number of logical registers (16 or 32)

```
mulsd   (%r8,%rax,8),%xmm0
add     $0x1,%rax
cmp     %eax,%r12d
addsd   %xmm0,%xmm1
movsd   %xmm1,(%rdx,%rdi,1)
jne     400740 <main+0x190>
```

Large physical register files (100s to 1000s)



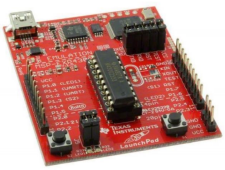
11

What about the HW do we need to know?

Personal Laptop/Desktop

Cloud Services

Microcontrollers



Accelerators

12

What about the HW do we need to know?

Out-of-order engine

Disk

Decoders

Prefetchers

ALUs

Program Counter

FPUs

Caches

I/O

Load/Store Units

AGUs

TLBs

DRAM

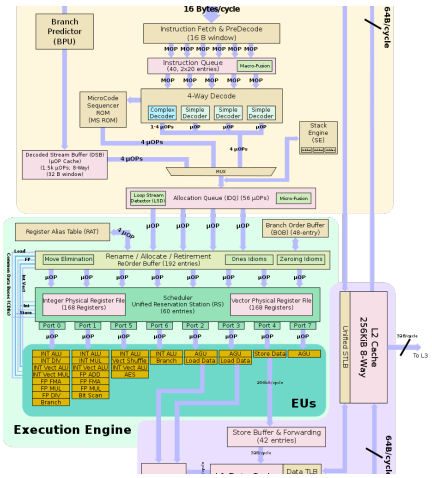
Registers

Branch Predictor

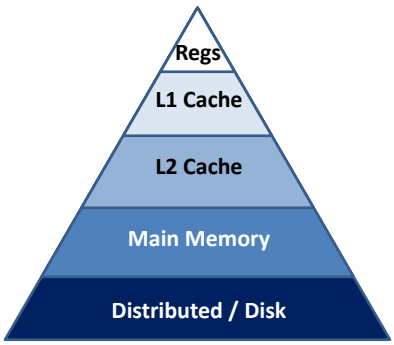
13

Two main parts – Execution & Data

Computation



Data Movement

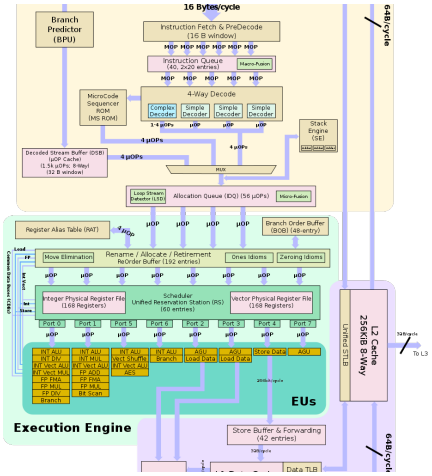


https://en.wikichip.org/wiki/intel/microarchitectures/haswell

14

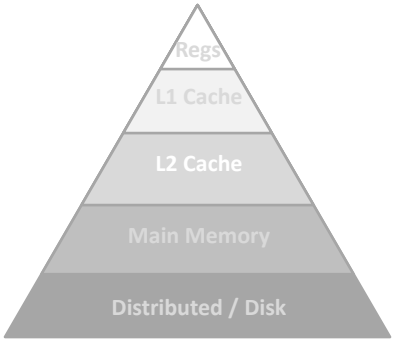
Two main parts – Execution & Data

Computation



https://en.wikichip.org/wiki/intel/microarchitectures/haswell

Data Movement



A short diversion

How to optimize this?

17

Scenario 1

Cashier's responsibility

- Scan Items
- Bag Items
- Check receipt / Distribute free gift

On Average:

- 2 min for each task

- Questions
  - How many customer (on average) every hour?
  - How long does each customer take?

18

Electrical & Computer ENGINEERING

You have been hired to speed up the supermarket checkout process

What are the different ways the checkout process can be improved?

- Explain why they help in speeding up the checkout process

How are they similar to hardware features we see in today's architecture?

19

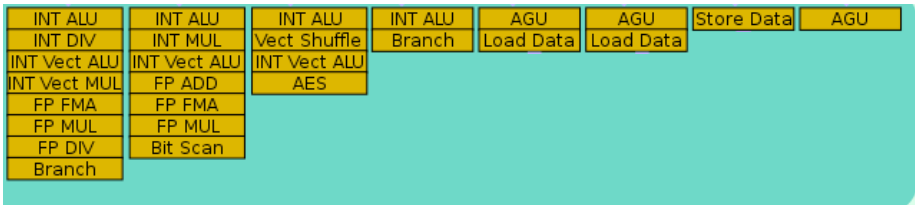
Electrical & Computer ENGINEERING

Back to architecture

20

Many functional units

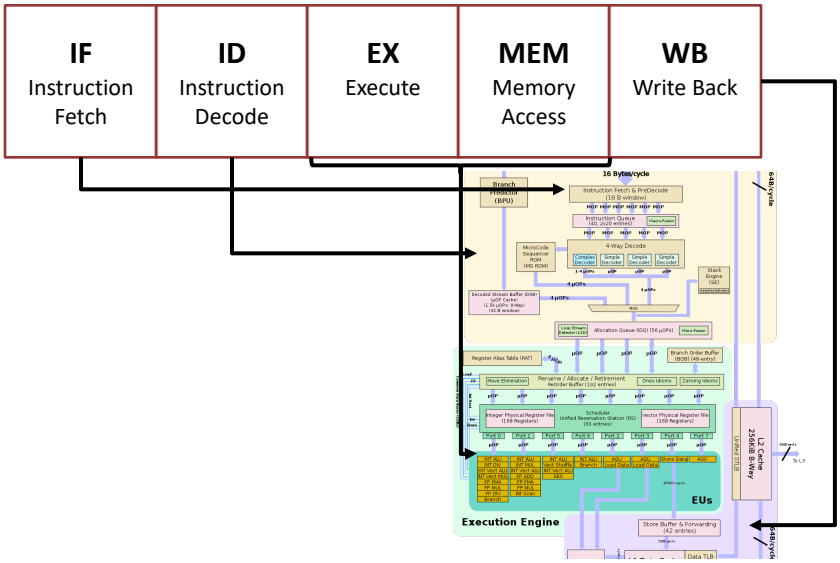
- Many queues/pipelines in hardware



- Goal of "Fast code" developer
  - Use as many pipelines as possible
  - Avoid stalling any of the pipelines

21

Model Instruction Pipeline



22

Instruction Pipeline

IF	ID	EX	MEM	WB
Instruction Fetch	Instruction Decode	Execute	Memory Access	Write Back

Most of the hardware pipeline are empty.
Fast code fills in the pipelines as much as possible

23

Instruction Pipeline

IF	ID	EX	MEM	WB
Instruction Fetch	Instruction Decode	Execute	Memory Access	Write Back





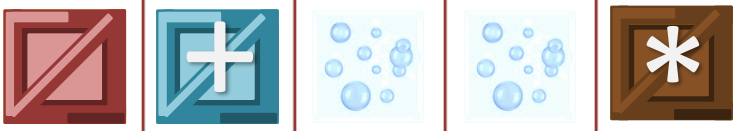
Different instructions can be in different stages

24

Instruction Pipeline

```
c += a*b;
```

IF	ID	EX	MEM	WB
Instruction Fetch	Instruction Decode	Execute	Memory Access	Write Back



Single dependence (*) can lead to multiple stalls (bubbles) in the pipeline. Drop in performance

26

Instruction Pipeline

```
c += a0*b0;
c += a1*b1;
```

IF	ID	EX	MEM	WB
Instruction Fetch	Instruction Decode	Execute	Memory Access	Write Back

Multiple instructions may still not completely eliminate bubbles in the pipeline.

27

Instruction Pipeline

```
c += a*b;
```

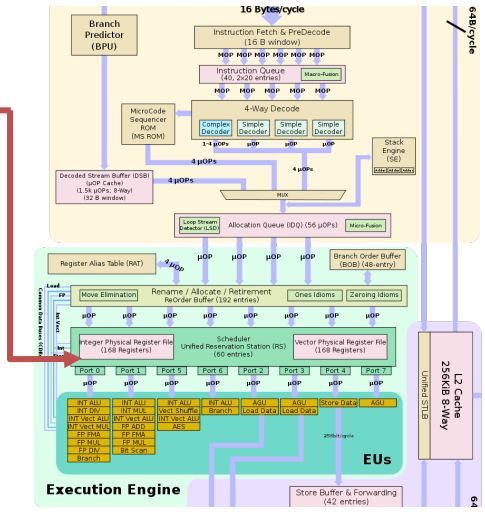
IF	ID	EX	MEM	WB
Instruction Fetch	Instruction Decode	Execute	Memory Access	Write Back

Need to find independent instructions!
Reorder instructions!

29

Instruction (Re)scheduling

- Hardware solution
  - Out-of-order Execution
    - Instruction buffer
    - Fire instruction when data is available
    - Write-back stage needs to be managed to ensure correctness



30

Instruction (Re)scheduling

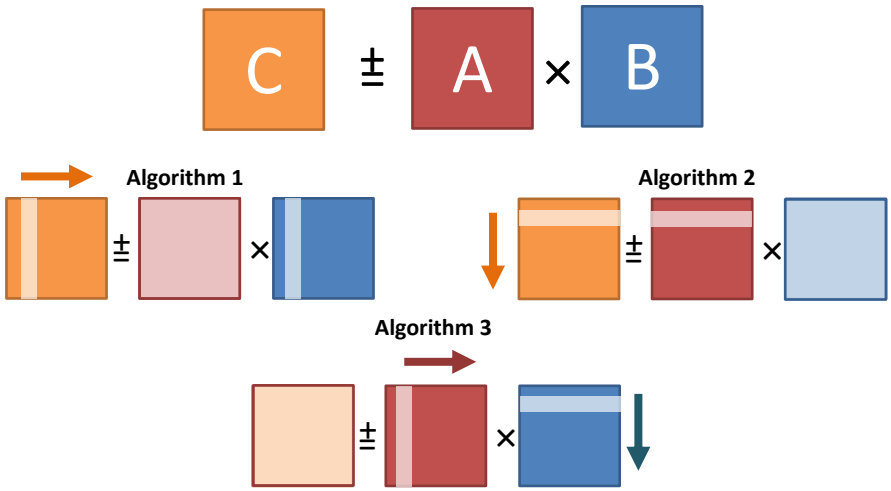
- Compiler solution
  - Software pipelining
  - Compiler tries to identify and moves the instructions during the compilation process

```
mulsd   (%r8,%rax,8),%xmm0
add     $0x1,%rax
cmp     %eax,%r12d
addsd   %xmm0,%xmm1
movsd   %xmm1,(%rdx,%rdi,1)
jne     400740 <main+0x190>
```

Remember `objdump` ?

32

Recap: Matrix Multiply as an Example



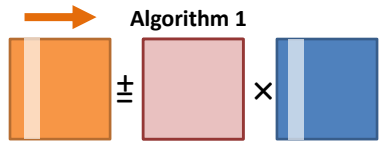
All $O(n^3)$. Which Algorithm?

33

Recap: Implementations of MMM

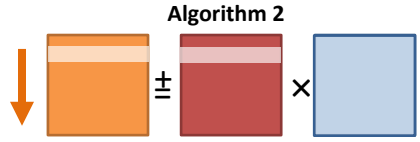
Algorithm 1

```
for (int j = 0; j!= n; ++j)
  for (int i = 0; i != m; ++i)
    for (int p = 0; p != k; ++p)
      c[j*m+i] += a[p*m+i] * b[j*k+p];
```



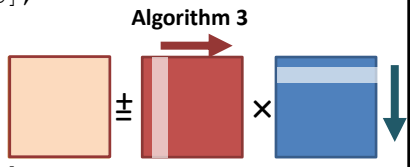
Algorithm 2

```
for (int i = 0; i != m; ++i)
  for (int j = 0; j!= n; ++j)
    for (int p = 0; p != k; ++p)
      c[j*m+i] += a[p*m+i] * b[j*k+p];
```



Algorithm 3

```
for (int p = 0; p != k; ++p)
  for (int i = 0; i != m; ++i)
    for (int j = 0; j!= n; ++j)
      c[j*m+i] += a[p*m+i] * b[j*k+p];
```

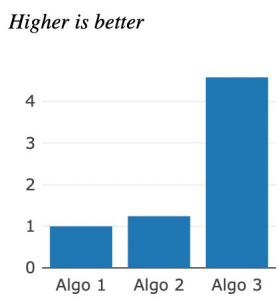


34

Matrix Multiply as an Example

- Using mmm.html

Loop Order	Execution Cycles
i, j, k	93 cycles
j, i, k	71 cycles
k, i, j	18 cycles



Why?

Hint: **Consider what is the basic sequence of operations for each algorithm**

35

Summary

- ISA is the “lowest” level exposed to the programmer
- ASM does not mean performance
- Instruction rescheduling is and **MUST BE** implemented at HW, SW and algorithm level
- Algorithms must be matched to hardware features