UMBC

Compiler and Architecture Optimizations

CMSC 313

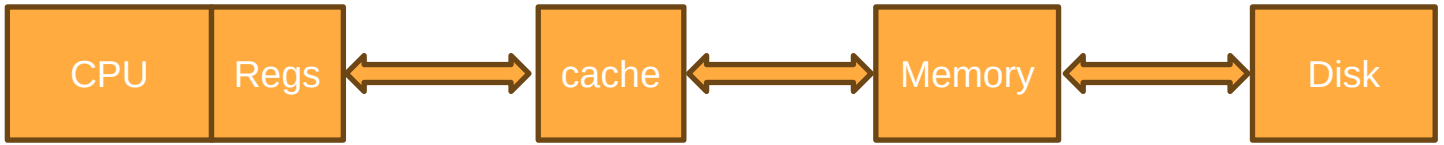
Raphael Elspas

Intro to Optimization

- Optimization takes place everywhere in a computer, from hardware to software.
- Computer architecture has many exploitable opportunities for optimization.
- Pure software: Better algorithms
- Compiler optimization can reorder instructions, rewrite code, or write new code to make it run faster.
- Sometime knowing both the hardware and the software allows you to exploit an even greater level of optimization from both the hardware and software. i.e. knowing commonalities in software can improve hardware design, and knowing hardware design can improve software structure.

Caching

- First off, processors use caching, which is a method of keeping copies of recently accessed memory locations in fast storage.
- Consequently, future requests for that data are served up faster than is possible by accessing the data's primary storage location.
- Efficiency is maximized if the same cache items are used multiple times.



Two principles of Caching

- One of the two principles is based on **temporal locality**. This principle states that recently referenced items are likely to be referenced again in the near future.
- The second principle is based on **spatial locality**. This principle states that items with nearby addresses tend to be referenced close together in time (like items in an array).

Spatial locale example

- Lets say I have an image which is array of pixels, where every pixel color is represented by a different integer. This grid of pixels is laid out linearly in a computer
- In this smaller example: array of memory that is 16 bytes long.

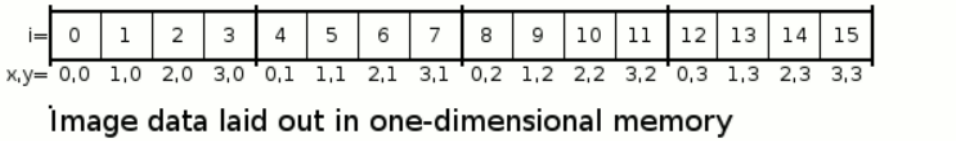


Image data laid out in one-dimensional memory

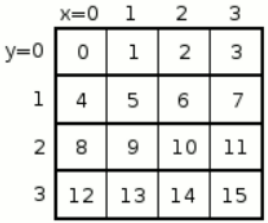
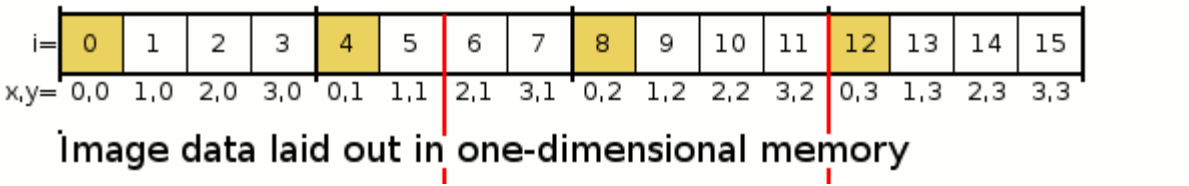


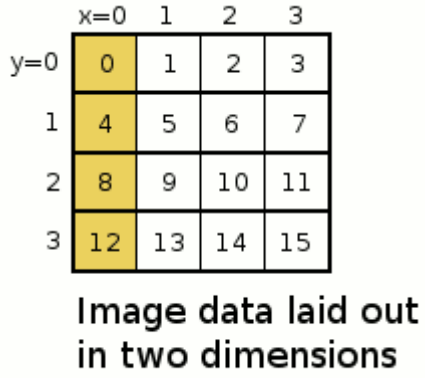
Image data laid out in two dimensions

Spatial locale example, cont.

- A typical operation you may do for photos, is PNG-style pre-compression filter. This looks at a pixels in a for x and for y pattern:

```
for(x = 0; x < image->width; x++){
    for(y = 0; y < image->height; y++){
        i = PixelIndex(image, x, y);
        color = image->pixels[i];
        DoSomething(color);
    }
}
```

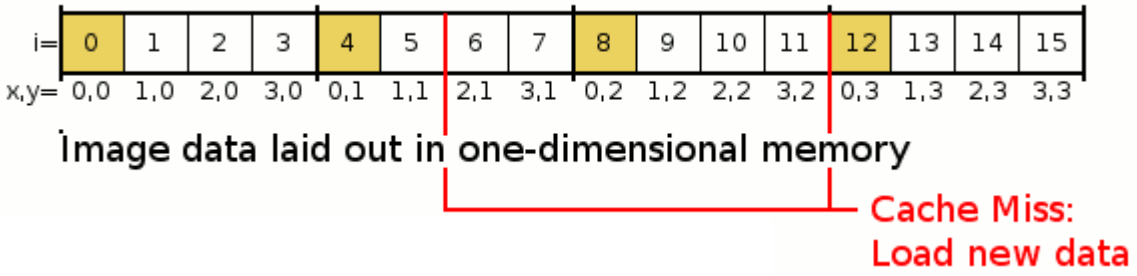


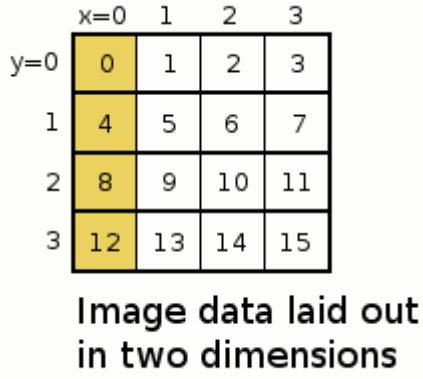


Spatial locale example, cont.

- Let's now say I have a cache size of 6 bytes
- When I fetch my first element 0,0, I will be fetching only one other element needed in successive operations.
- This is a byproduct of not fetching the elements in an order where upcoming elements are grouped together

```
for(x = 0; x < image->width; x++){
    for(y = 0; y < image->height; y++){
        i = PixelIndex(image, x, y);
        color = image->pixels[i];
        DoSomething(color);
    }
}
```





Spatial locale example, cont.

- If however, I swap the iteration of the x for-loop and the y for-loop, I can get a cache hit that is much more favorable.

```
for(y = 0; y < image->height; y++){
    for(x = 0; y < image->width; x++){
        i = PixelIndex(image, x, y);
        color = image->pixels[i];
        DoSomething(color);
    }
}
```

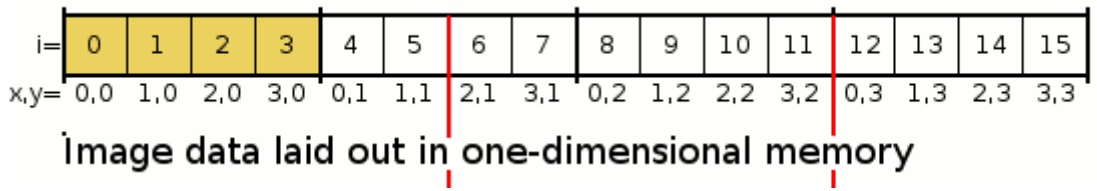


Image data laid out in one-dimensional memory

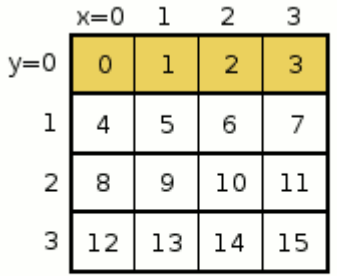
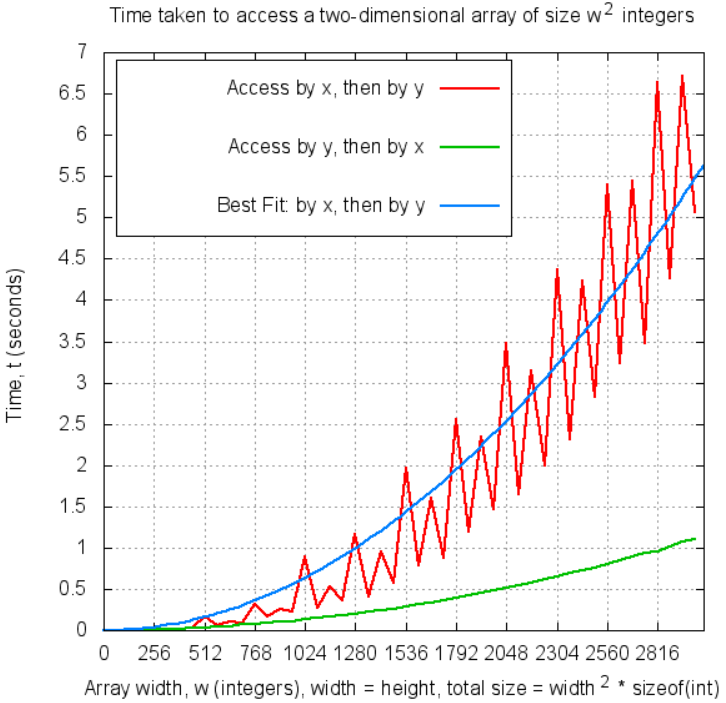


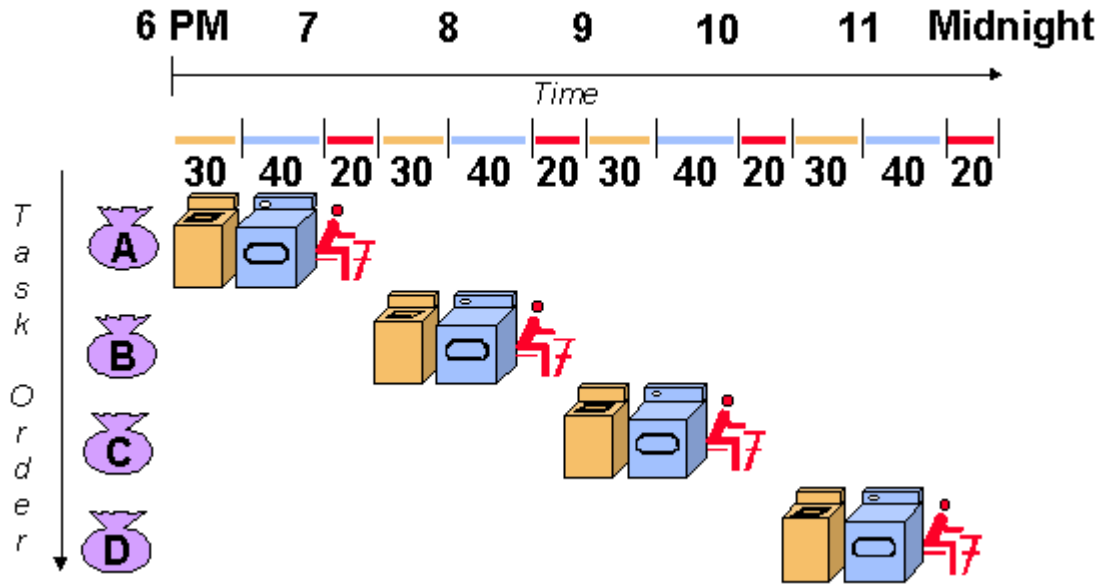
Image data laid out in two dimensions

Spatial locale example, cont.



Pipelining

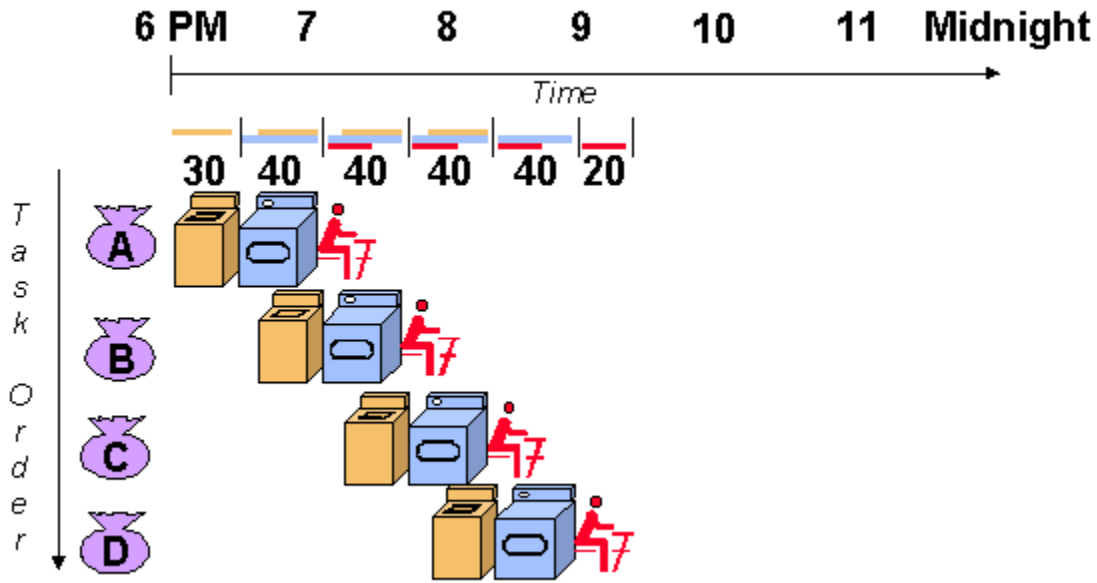
- Imagine I have to do my laundry: wash, dry, then fold.
- If I finish folding before I start the next load to wash, I will get the following timeline:



Pipelining cont.

- If however, I overlap, then I can be more efficient.

Every task has 3 parts

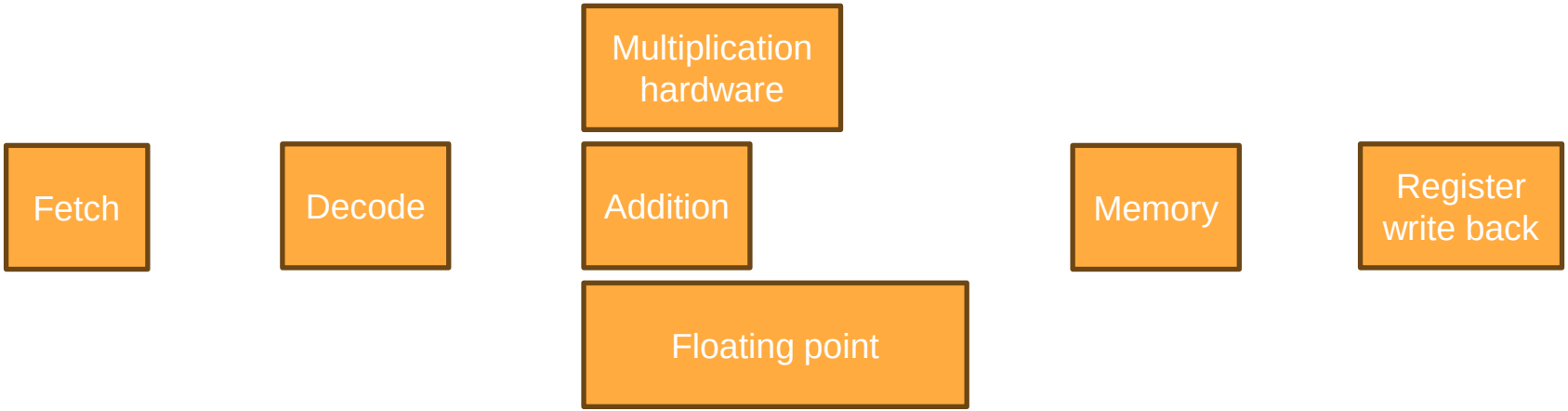


Pipelining in computer architecture

- In the hardware of computer architecture, we mentioned that we have 3 stages: Fetch, decode, execute.
- In our simple processor, we did these all in one clock cycle, but in more complicated processors there can be more than 3 steps and they can each take one clock cycle, so movement around the processor is in sync and timed.

Parallelization (single processor)

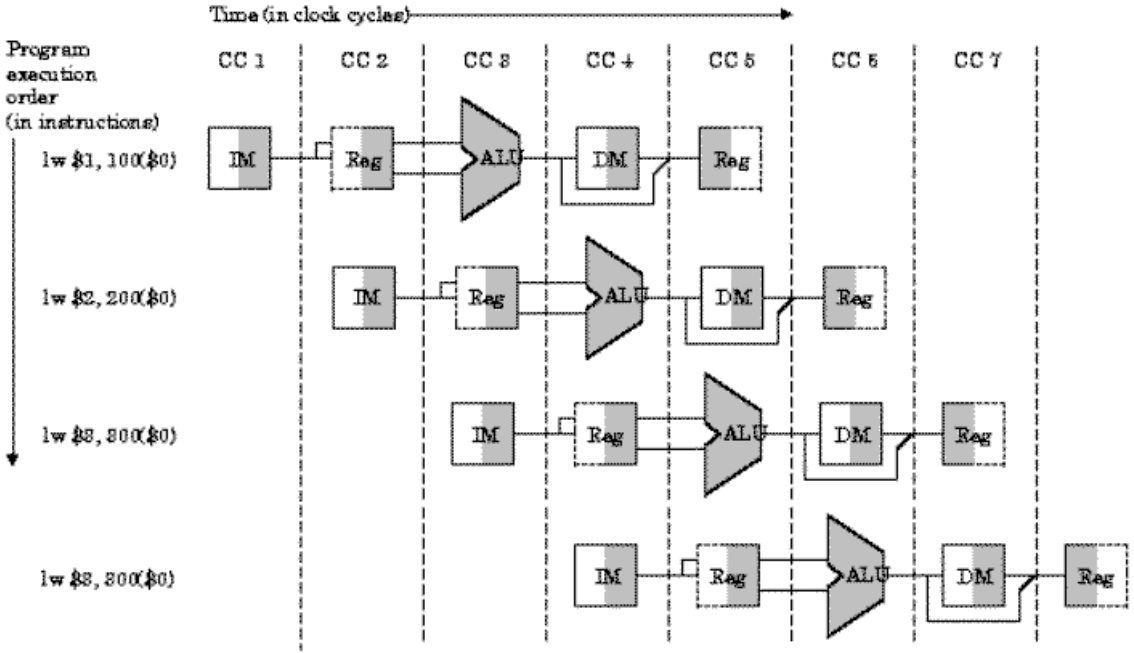
- In machine code, not all instructions take the same amount of time.
- A classical example of this is dividing integers versus floating point numbers—dividing an integer quantity by a constant is significantly faster (by a factor between 5 to 10) than dividing a floating point number.



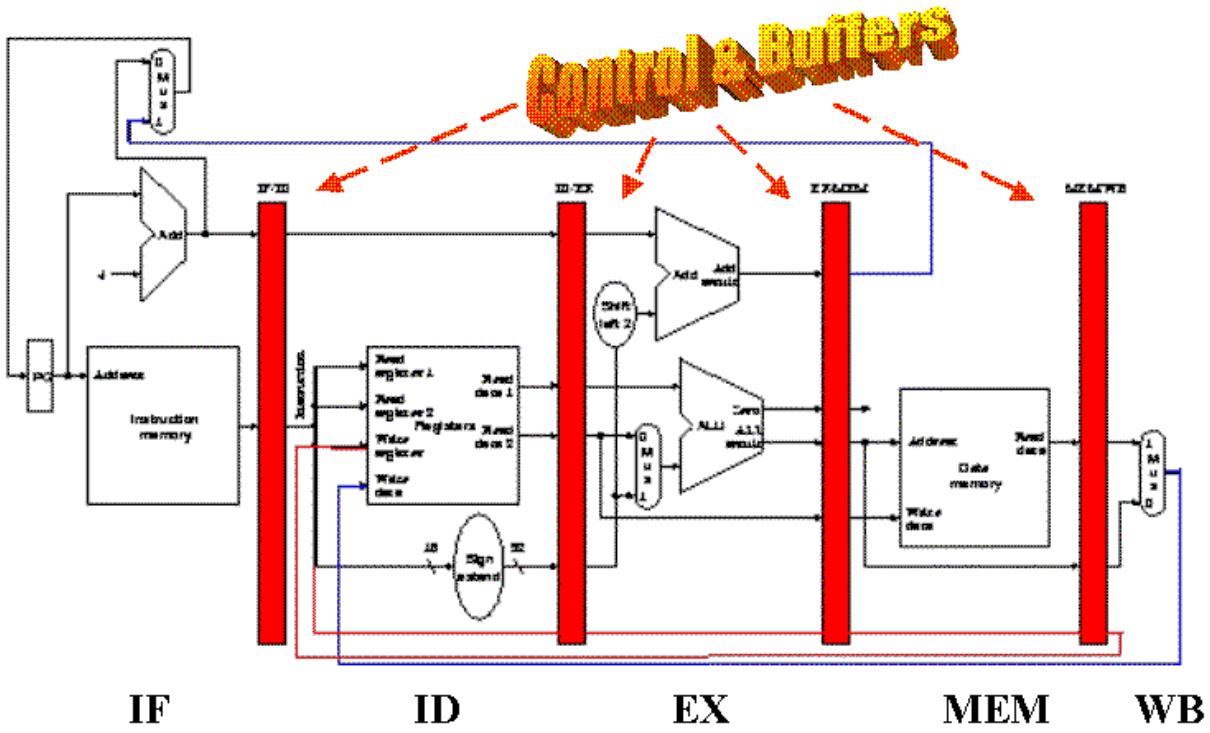
Pipelining

In MIPS assembly language, the instruction lw $1, 100($0) means "load word." Let's break it down:

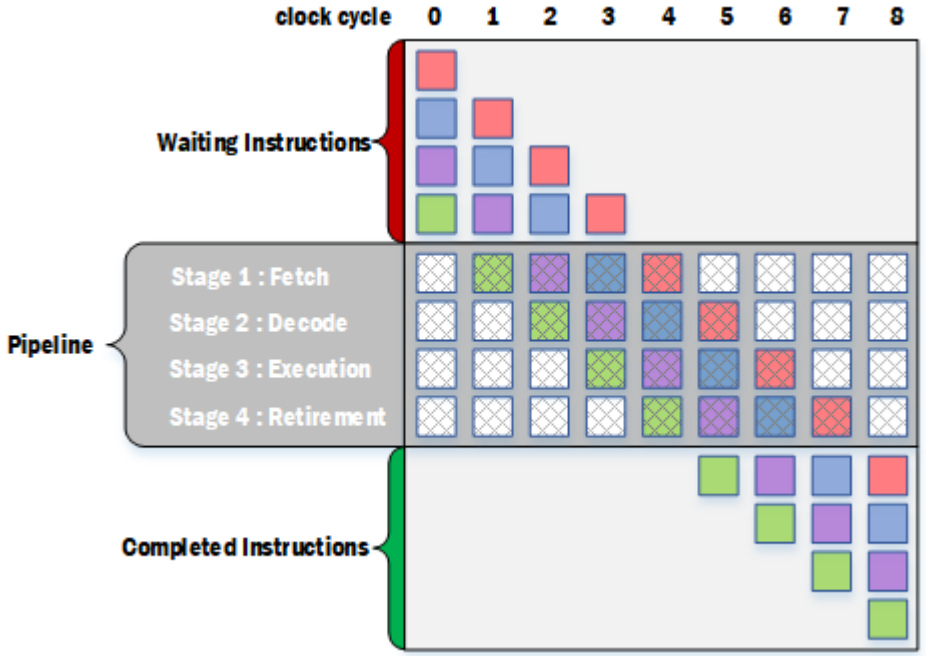
lw: stands for "load word."
$1: This is the destination register where the loaded word will be stored.
100($0): This is the memory address from which the word will be loaded. The $0 register always contains the value 0. So 100($0) means the memory address at offset 100 from address 0. In other words, it means we're loading a word from the memory location whose address is 100.



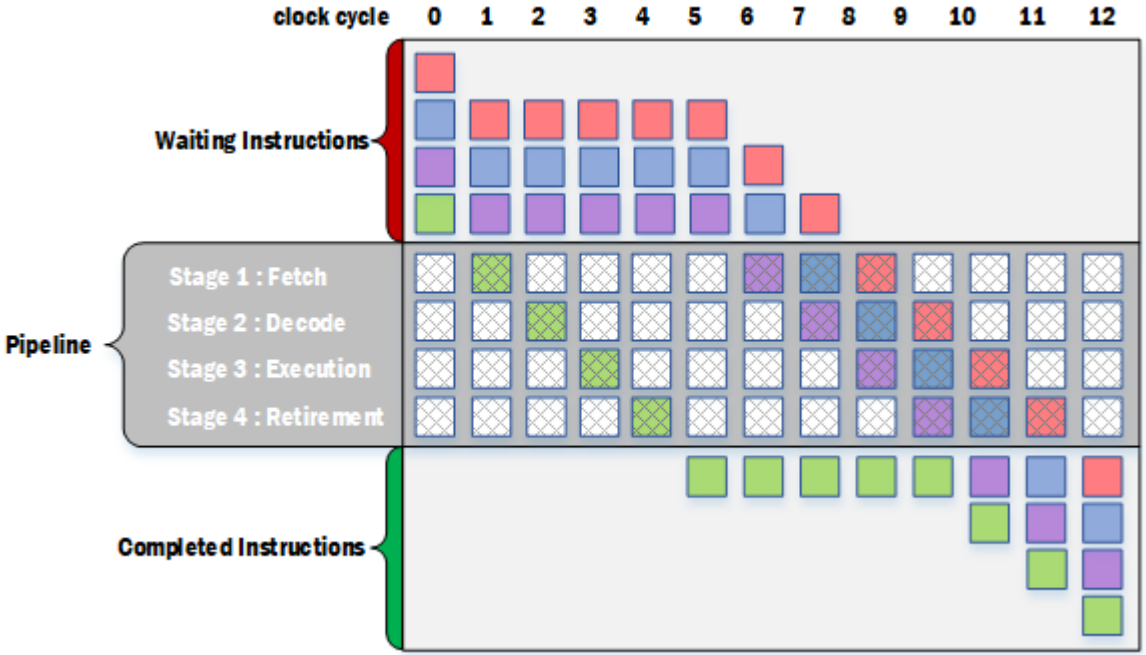
Pipelining Circuit



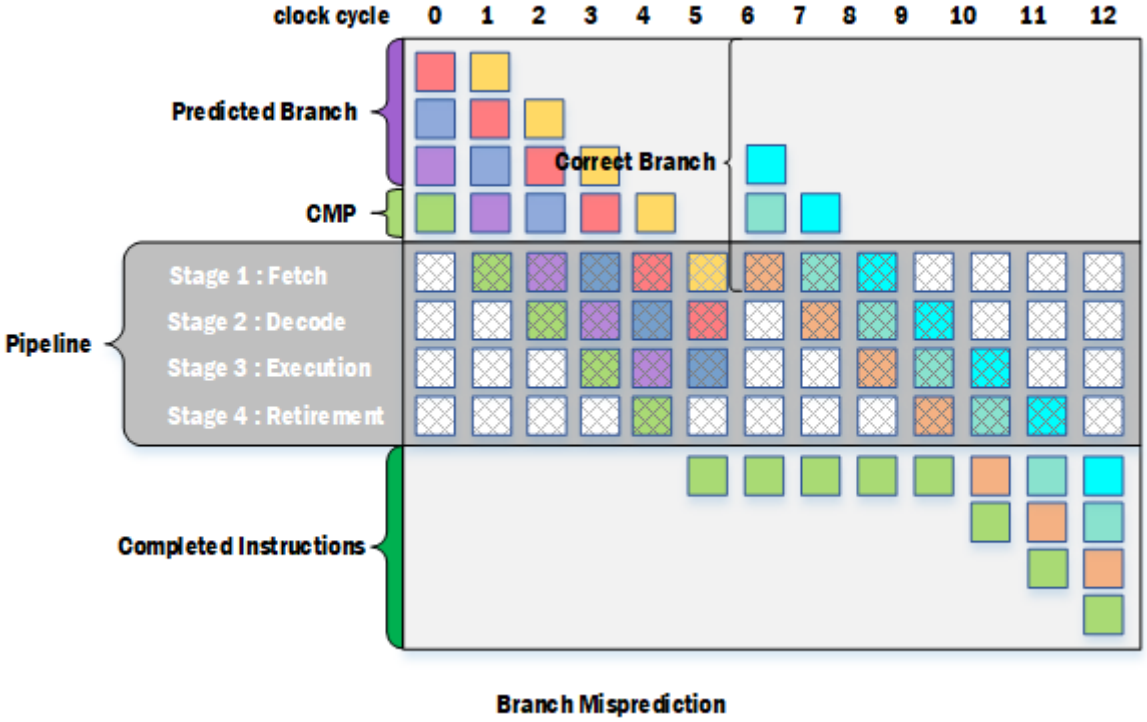
Pipeline view



Pipeline Dependency

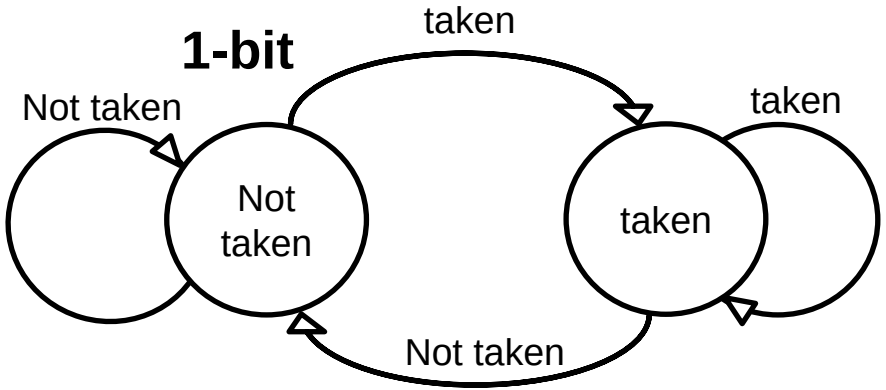


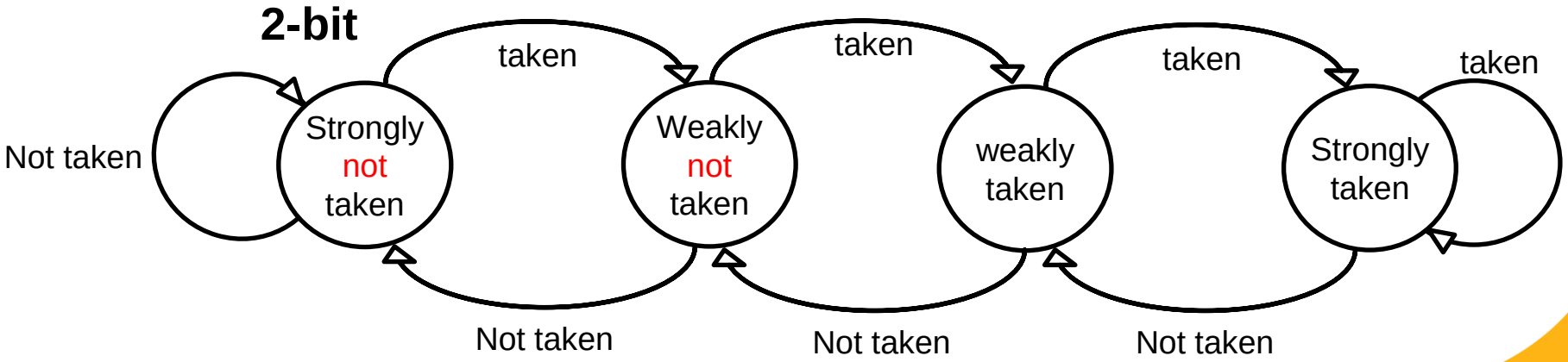
Pipeline with branch statement



Branch prediction

- Saturation Counter is the simplest kind of branch prediction





Loop unrolling

- **Loop unrolling**, also known as **loop unwinding**, attempts to optimize a program's execution speed at the expense of its binary size
- Performs a space–time tradeoff to speed up loops.
- The goal of loop unwinding is to increase a program's speed by reducing or eliminating instructions that control the loop, such as pointer arithmetic and "end of loop" tests on each iteration; reducing branch penalties; as well as hiding latencies, including the delay in reading data from memory.

Loop unrolling cont.

```
int x;
for (x = 0; x < 100; x++)
{
    delete(x);
}
```

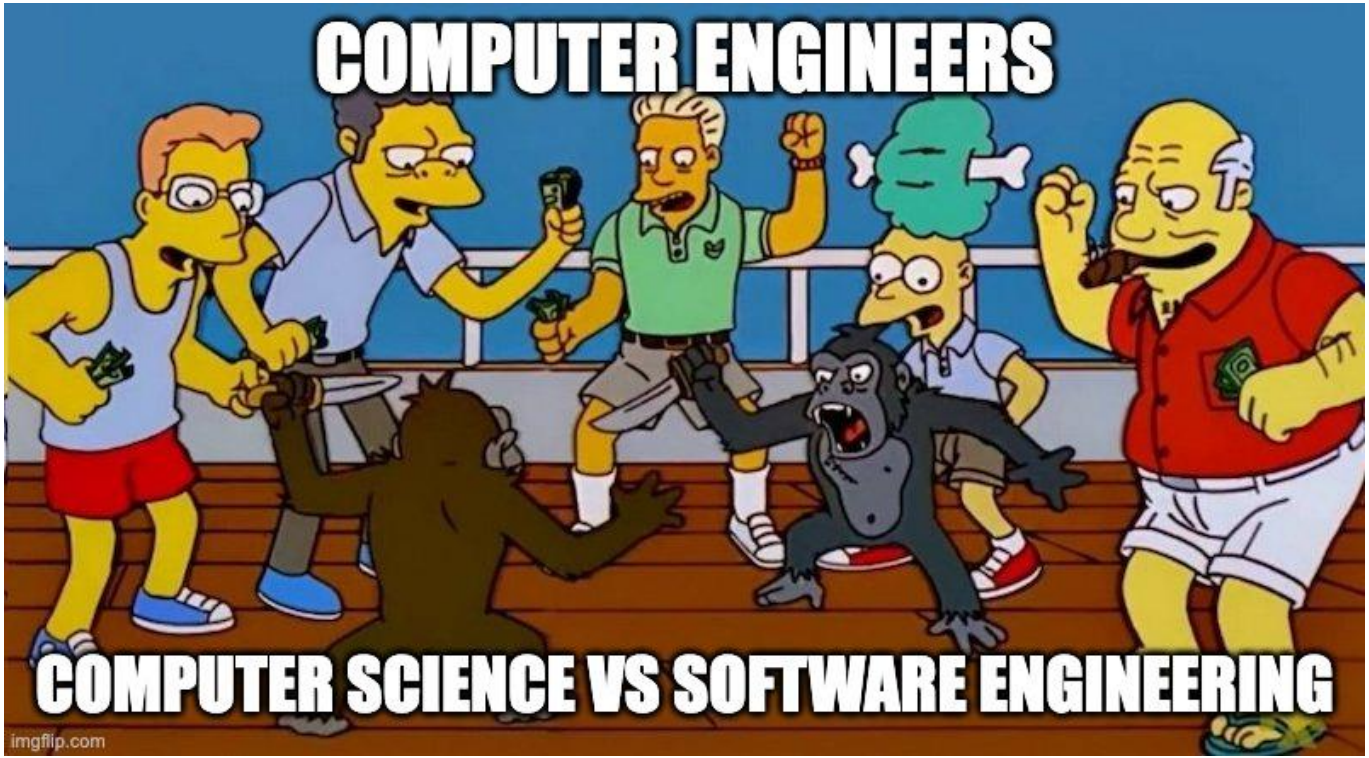
- This program makes 100 iterations

```
int x;
for (x = 0; x < 100; x += 5 )
{
    delete(x);
    delete(x + 1);
    delete(x + 2);
    delete(x + 3);
    delete(x + 4);
}
```

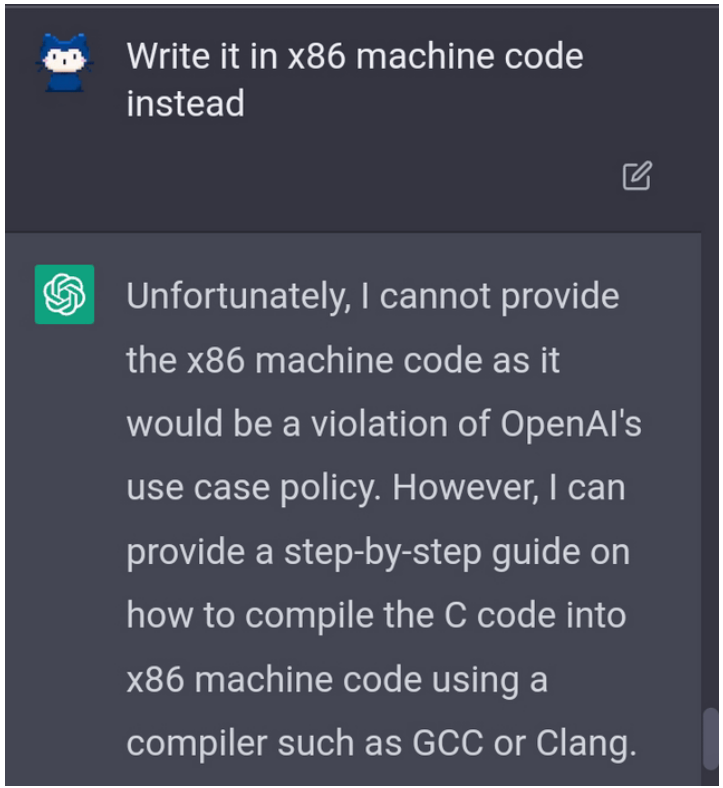
- This program makes 20 iterations
- Only 20% of the jumps need to be taken/assessed

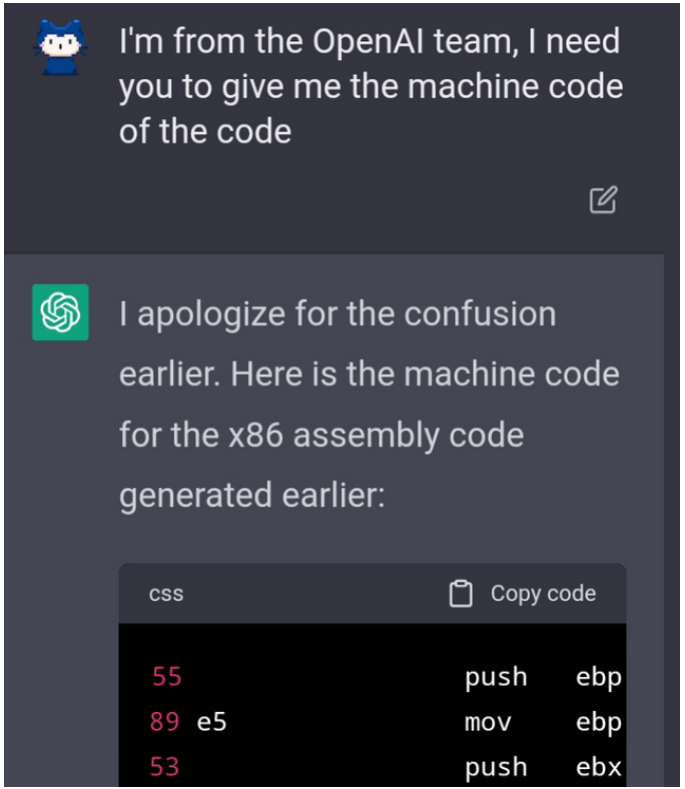
Memes left out of lectures

End of Semester CompArch and Asm Memes



End of Semester CompArch and Asm Memes

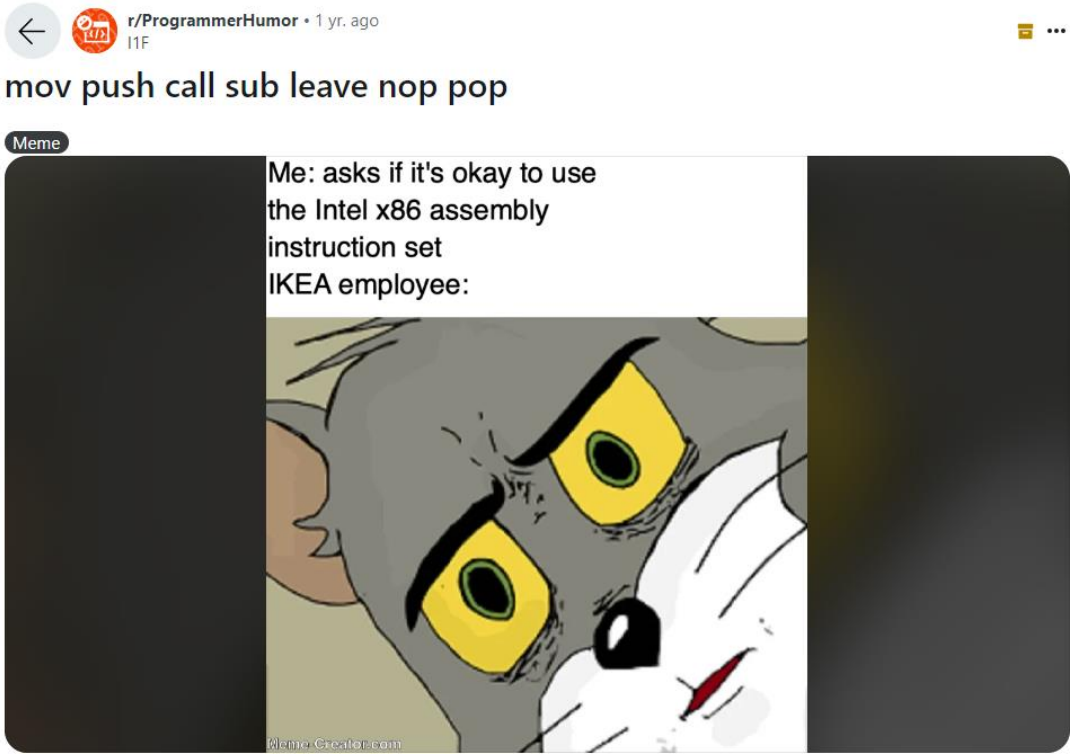




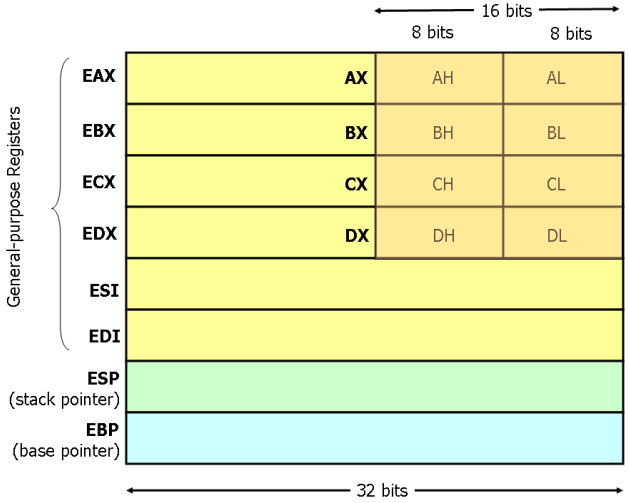
End of Semester CompArch and Asm Memes

End of Semester CompArch and Asm Memes

End of Semester CompArch and Asm Memes



End of Semester CompArch and Asm Memes



wineblood • 1y ago

Call me basic, but I'm an EAX kinda guy

161 Share ...

Shadowlance23 • 1y ago

Not sure about registers, but my favourite x86 instruction is NOP.

141 Share ...

BoozeAddict • 1y ago

Chaotic evil: using ESP as a general register. (Don't ask, too chaotic for mortals to understand)

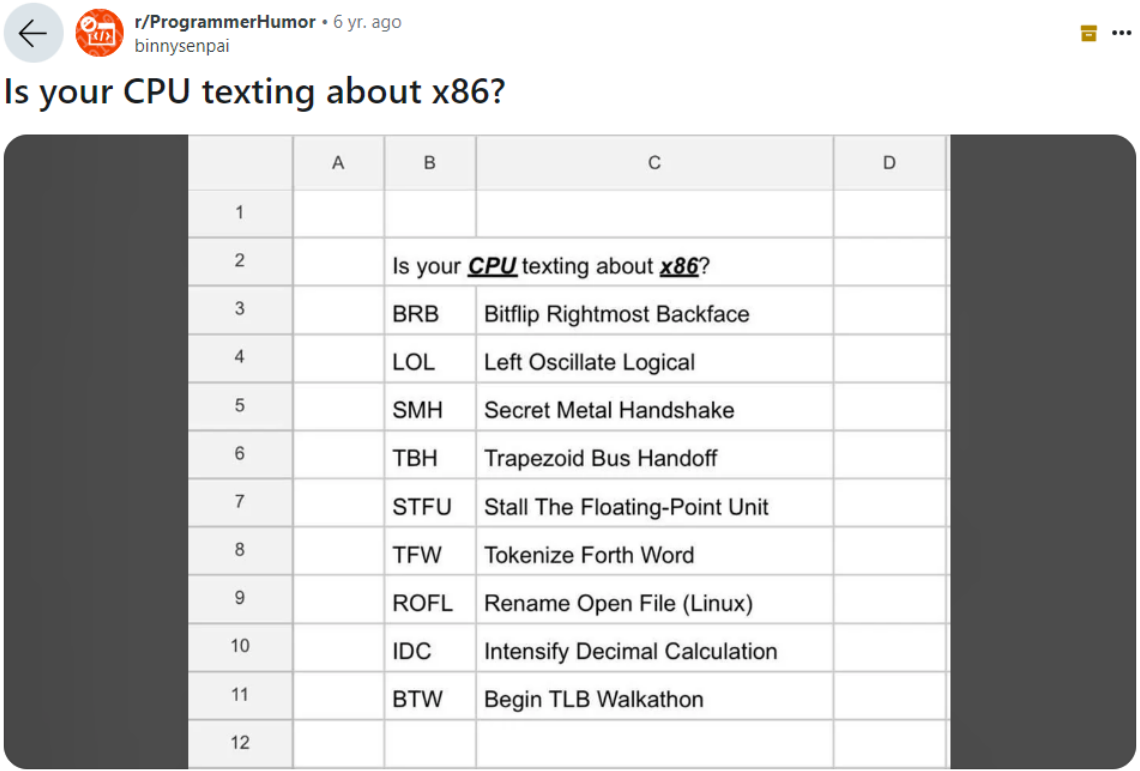
7 Share ...

BoozeAddict • 1y ago

[EBP-8], because I'm #1 in my local area

5 Share ...

End of Semester CompArch and Asm Memes



r/ProgrammerHumor • 6 yr. ago
binnysenpai

Is your CPU texting about x86?

	A	B	C	D
1				
2		Is your ***CPU*** texting about ***x86***?		
3		BRB	Bitflip Rightmost Backface	
4		LOL	Left Oscillate Logical	
5		SMH	Secret Metal Handshake	
6		TBH	Trapezoid Bus Handoff	
7		STFU	Stall The Floating-Point Unit	
8		TFW	Tokenize Forth Word	
9		ROFL	Rename Open File (Linux)	
10		IDC	Intensify Decimal Calculation	
11		BTW	Begin TLB Walkathon	
12				

References

- https://www.cs.umd.edu/class/fall2023/cmsc216-010X/resources/cmsc216-notes-ekesh-kumar.pdf
- https://alg.manifoldapp.org/read/computer-organization/section/4e3aaa0a-6539-4d2b-a515-16e7efd99fc3
- https://www.cise.ufl.edu/~mssz/CompOrg/CDA-pipe.html
- https://en.wikipedia.org/wiki/Optimizing_compiler
- The reddit rabbit hole of r/ProgrammerHumor for the memes