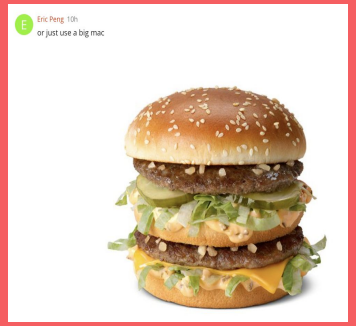
Welcome back to CS999H!

Week 9

Ed meme recap:

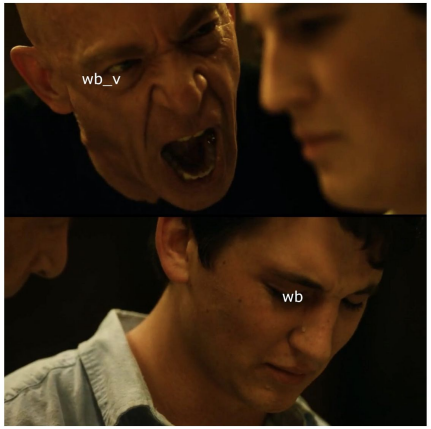


"WERE YOU FLUSHING OR WERE YOU STALLING" #474

Anonymous
3 days ago in Assignments – P7

PIN STAR WATCH 146 VIEWS

7



I give up too buddy #466

Sohan Subudhi
7 days ago in General

PIN STAR WATCH 167 VIEWS

9

```
ssubudhi@apple-jacks:~/cs429/p7_practice$ iverilog cpu.v -o cpu
cpu.v:95: syntax error
cpu.v:98: error: invalid module item.
cpu.v:101: syntax error
cpu.v:105: error: invalid module item.
cpu.v:106: syntax error
cpu.v:110: error: invalid module item.
cpu.v:111: syntax error
I give up.
```

We give up.

Questions on lecture content?
Or about cats?

Stress

- 429H is not an easy class
  - Lots of new materials
  - Unfamiliar programming environments
  - Fast, often relentless pace
- Struggling in this course is normal
  - There will be times you won't know the answer of the solution
  - This is expected—we want we everyone to succeed, but the only way we can help is if you ask for it
- If you find yourself overly overwhelmed or spending more time on this class than you think you should be, please reach out to Dr. Gheith or the TAs
  - We can help out as far as the class goes
  - We can provide other resources where we are not able to help

Mental health resource available at UT

review

Quiz everyone say CHEESE!

Question 1

hill.c (In Chrill's favorite font)	hill.ok

#include <stdio.h> // include standard io library
#include "src/go.h" // include go implementation
Channel *ch; // defines global channel pointer
Value f1() {
  printf("1"); // print 1
  Value v = recv(ch); // receive the 429
  printf("2"); // print 2
}
int main() {
  ch = channel(); // make a new channel, assign to ch
  go(f1);
  printf("a"); // print a
  send(ch, 429); // send the 429
  printf("b"); // print b
  printf("c"); // print c
  return 0; // return from main
}
	1a2bc
	
	Other Possible Outputs
	
	a12bc
	1abc
	a1bc

Question 2

malloc(3);
malloc(3);
malloc(4);
malloc(4);

Question 3

Any questions on parts of this question?

3. **[5 pts; 15 mins]** The fib8 architecture has:

8 general purpose registers (r0, r1, r2, ..., r7), each 8 bits wide and a PC of 8 bits
Byte-addressable memory
Fixed length instructions
0 cycle latency for registers and memory

Here is the encoding of its instructions:

ins	encoding	description
swp	00aaabbb	swap(reg[aaa], reg[bbb])
js	01aaabbb	if (reg[bbb] < 0) pc = reg[aaa]

(note that `js` treats reg[bbb] as a signed value)

You are allowed to use D flip flops, muxes, decoders, an adder, and any multi-input logic gates. Any other abstractions need to be defined before you can use them.

Implement a processor that supports the fib8 architecture. You can use a register file that has two read ports and two write ports and a main memory module that has one read port.

Final Project

Final Project Info!

- work in groups of **up to four people**
- presentations will be April 25th and April 26th
- anything architecture related
  - extend a project we already did
  - something completely new
- a project proposal will be due at some point lol :D
- form groups + ideas **next week**

What we are looking for in presentation

- Be prepared!!
  - Have a backup plan if your live demo doesn't work
- Explain your work
  - Provide background that is appropriate for CS429H students
  - Ideally people will learn something about architecture from your presentation!
- Demonstrate what you did
  - Show screenshots of results, live demos, whatever is appropriate for your project

Final Project Ideas !!!

- We will post a long list of project ideas on Ed
- Note: We have 2 FPGAs (maybe more) so please let us know early if you'll want one!

P7

Poll

How's your status on P7?

A. What's P7?
B. I've heard of it
C. I've cloned the starter code and/or looked through it
D. I've started planning/writing code
E. I'm mostly done but might still have bugs
F. P7 any% speedrun

Arts & Crafts

We need three volunteers, you will be creating the next big masterpiece on the whiteboard

Arts & Crafts

We need three volunteers, you will be creating the next big masterpiece on the whiteboard

You get: 3 dry erase markers

We want: the drawing to the right repeated 25 times on the board and we want it done fast

The tree, tent, and stars+moon should each be a different color

As a class, you have 5 minutes to decide the fastest way to draw this mural

Single Cycle Processor

- One person at a time
- Have to wait for order to complete before anyone else is allowed to begin
- If each order takes T time to complete, then N orders take N*T time

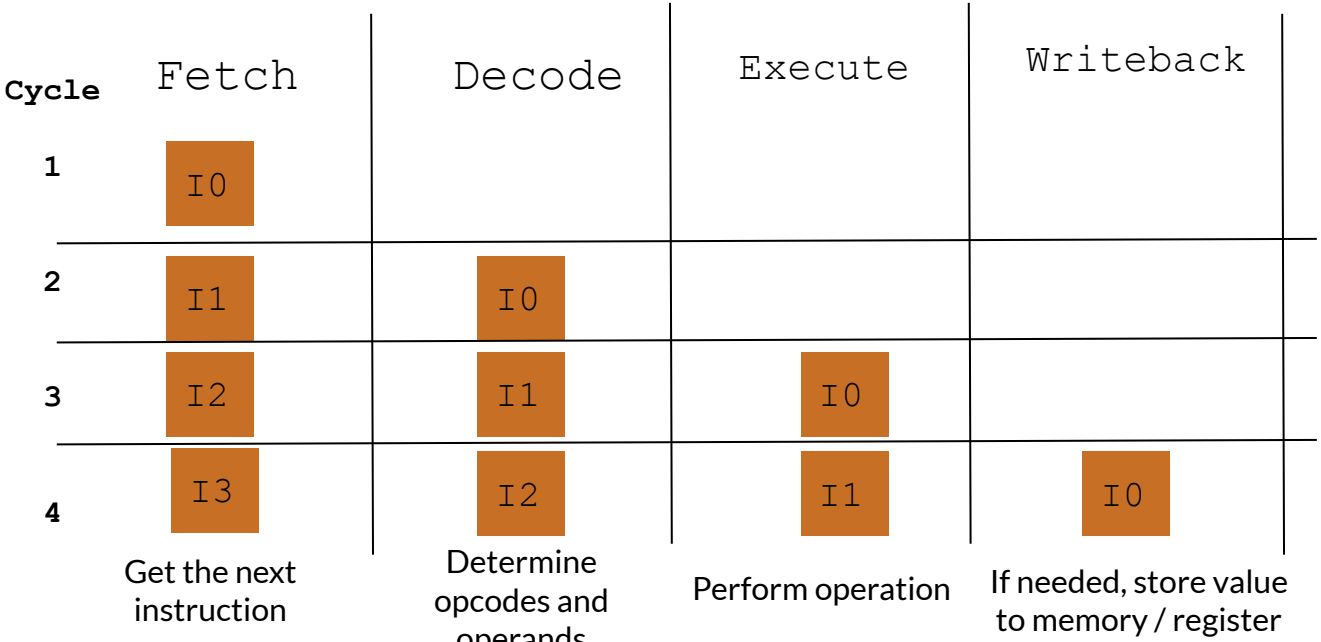
Pipelined Processor

- L people at a time
- While one person is paying for their food, another can start making their burrito
- Each order takes T time, but now N orders takes ~N*T/L time since we keep the pipeline full

Memory Latency

- RAM in real computers usually takes ~100 cycles to access
  - Wire length matters! The speed of light isn't that fast
  - Caches usually range from ~3-20 ish depending on level
- How does this affect your processor?
- What happens when you want to use a value that gets loaded from memory?

Pipeline Stages



We need to figure out:

- How to ensure the right information is passed from beginning to end
- How to account for memory and register read/write delays
- How to handle hazards (next slides)

Hazards

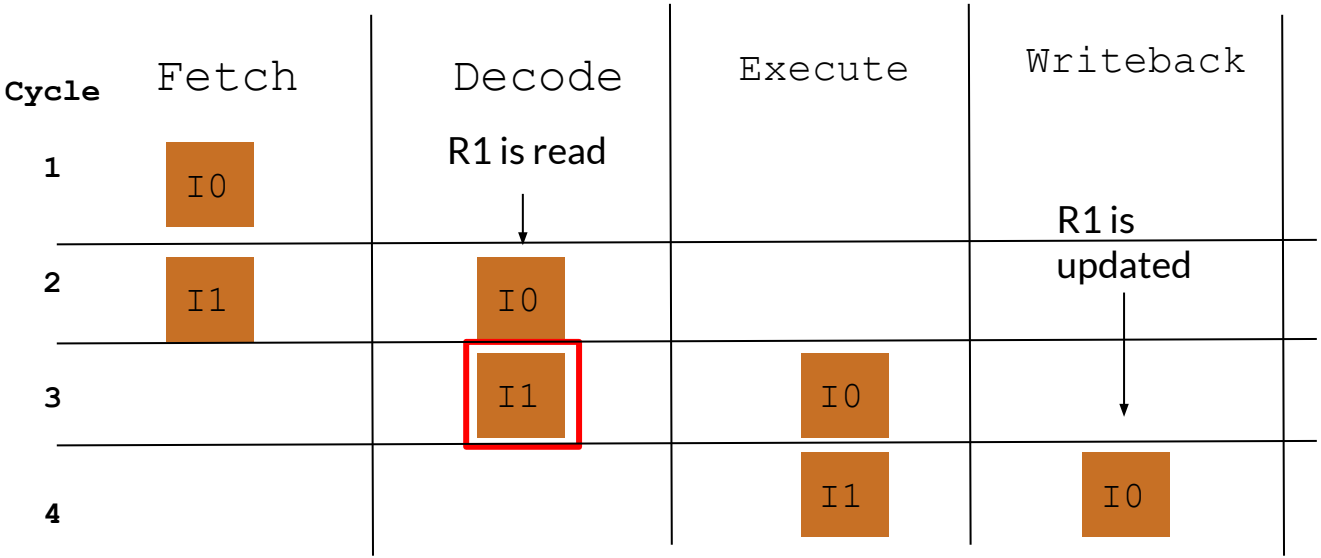
- What is a hazard?
- What are the three different types of hazards?

Hazards

- Prevent instructions from executing one after the other
- **Data hazards**
  - When a later instruction depends on the result of a previous instruction
- **Control hazards**
  - When instructions change the PC
- **Resource hazards**
  - When hardware can't support certain instructions happening simultaneously
- Solutions:
  - (Often) forces you to insert cycles between certain instructions
    - Stalling, flushing
  - Forward values to where they need to go

Data Hazards

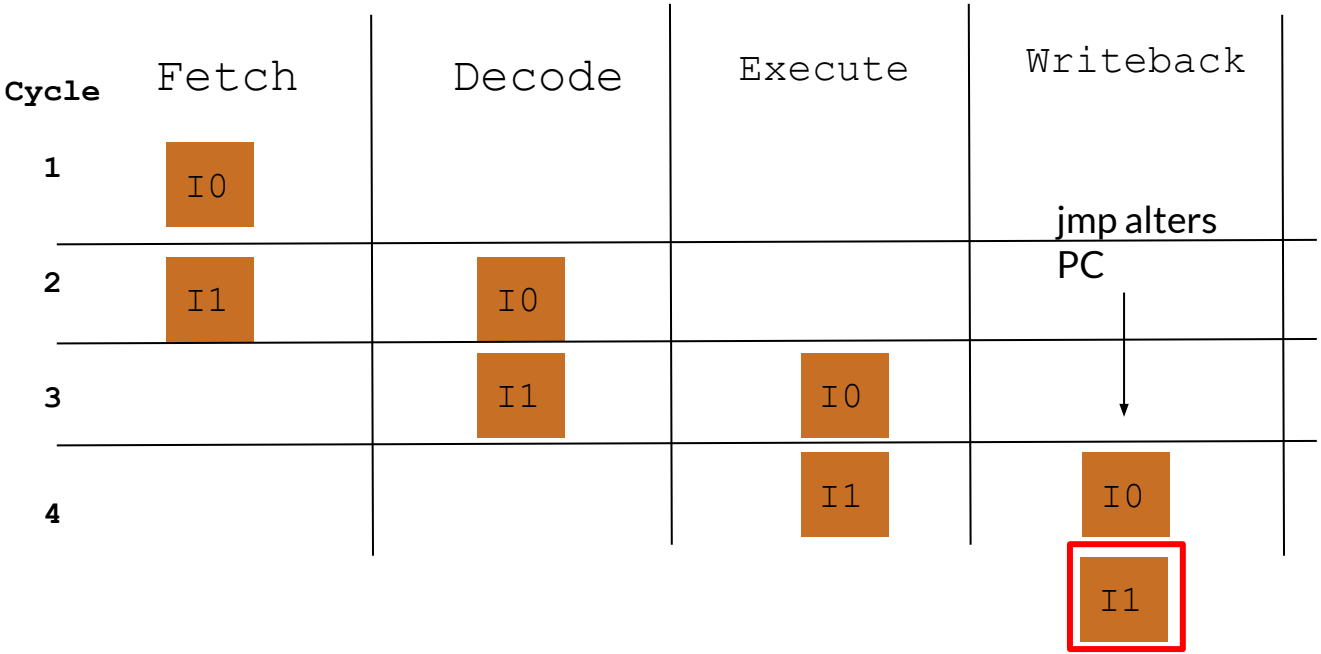
```
I0: movl r1, 3
I1: sub r2, r1, r0
```



r1 won't be updated when i1 reads its value from the register file

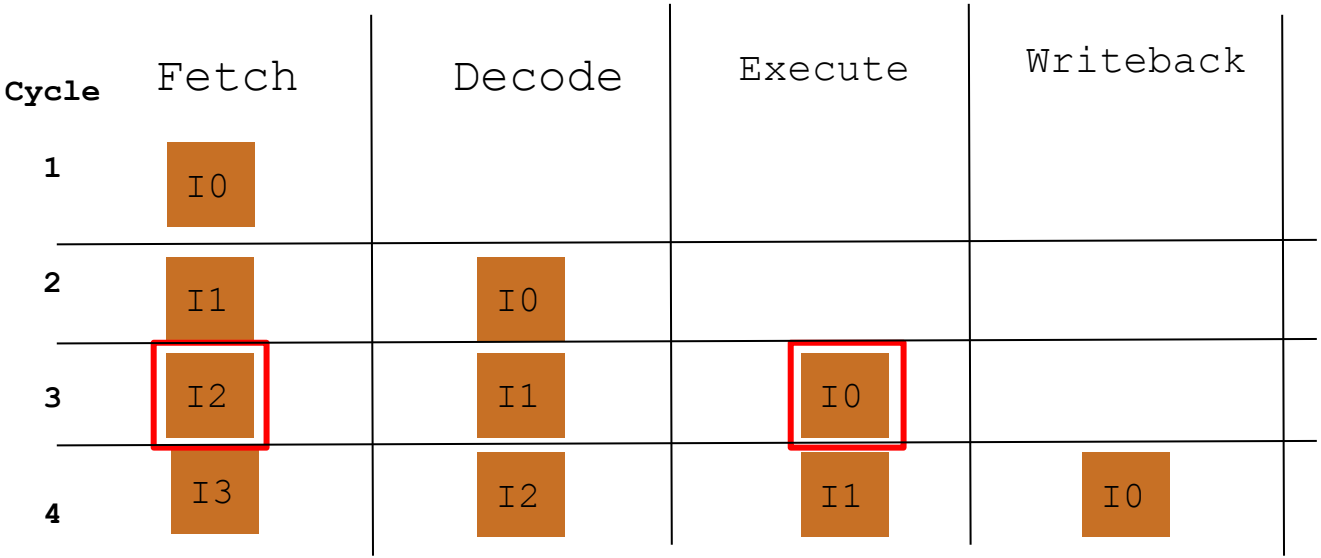
Control Hazards

```
I0: jmp r1
I1: *something*
```



i1 should not get committed since we branch to another instruction

Resource Hazards




```
I0: ld r1, r2
I1: *something*
I2: *something*
I3: *something*

What could go
wrong if we only
had one read port
for mem?
```

Can't fetch ins + load at the same time

A Real Processor

- https://en.wikipedia.org/wiki/List_of_Intel_CPU_microarchitectures
- http://users.utcluj.ro/~baruch/book_ssce/SSCE-Intel-Pipeline.pdf
- Modern Intel processors have 14 pipeline stages
- Split into the same basic chunks as ours!

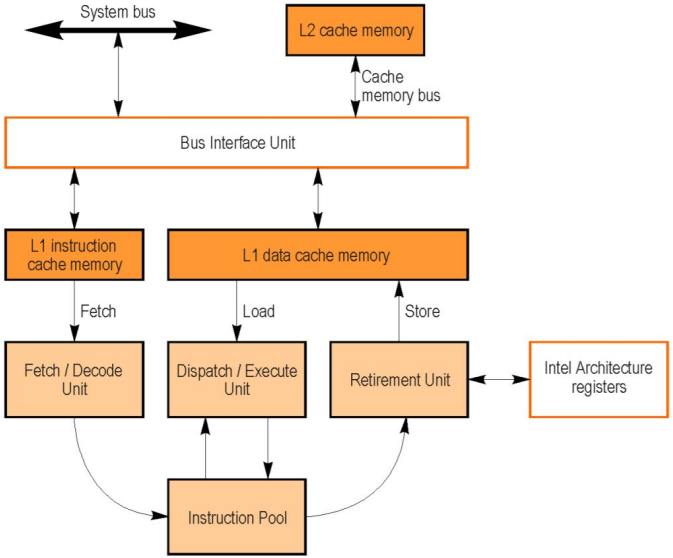
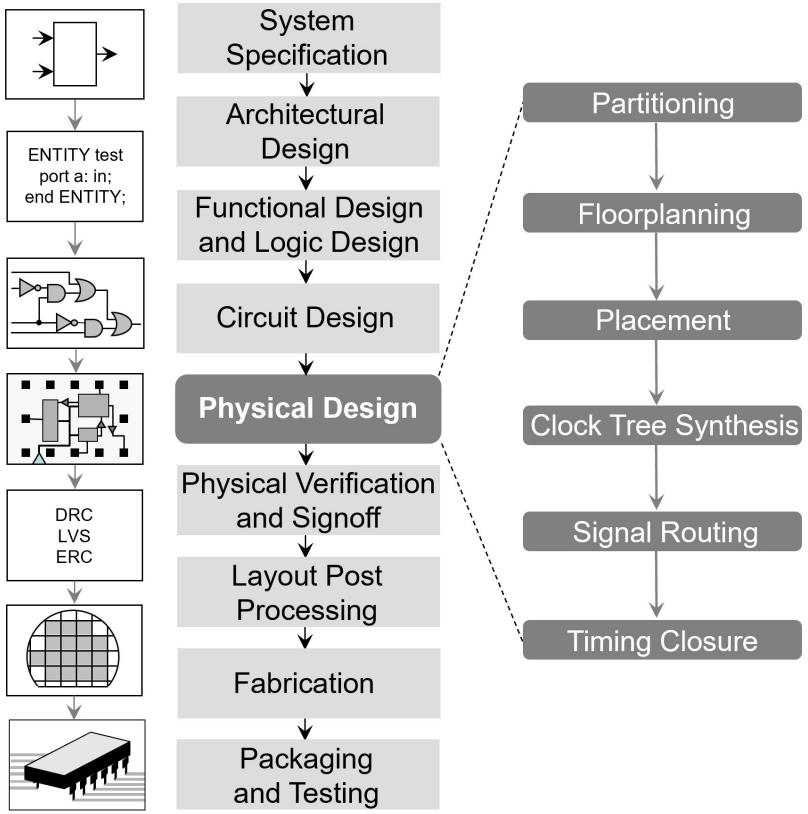


Figure 5.17. Conceptual view of the Intel Architecture processors pipeline.

The Hardware Development Cycle

- What is Verilog?
  - A description of hardware interactions
  - Only a small piece of hardware development



Verilog Advice

Style Guide

Continuous Block

- Only use =, <= is a syntax error
- Always declare wires/regs as [x:0], not [0:x]

Procedural Block

- Only use <=, NEVER USE =
- Only use always @(posedge clk), don't use negedge or other things in the @()
- Every line is run at the same time, so you can swap values like this:

```
always @(posedge clk) begin

    rA <= rB;

    rB <= rA;

end
```

Tips

- Follow the style guide
- Follow the style guide
- Don’t $display debug
- Add -Wall to the iverilog compile options
- Don’t touch verilog functions unless you know what you’re doing
- Ignore hazards, flushing, stalling etc. to start, and slowly add those in
- Use good wire & reg naming conventions (know which things are inputs to your stage and what are outputs)
- Clearly mark and separate each stage with some consistent convention
- You can use multiple procedural blocks
- Your test case (a .hex file) MUST HAVE COMMENTS
    - It is fine to share assemblers/disassemblers, but your test case should be pretty understandable without having to use a disassembler

some advice for debugging

- check that there are no blocking statements in your always blocks
- all always blocks should be **@ (posedge clk)**
  - (except for in the clock module)
  - you **will get a 0** if you do not follow this
- if you have an if statement in an always block, are you updating the same set of registers in both the if and the else? If not, is it intentional?
- are you updating the same register in multiple locations?
- the memory and register modules **cannot** stall
- **a correct implementation that flushes on every hazard will get more correctness points than an incorrect implementation that attempts stalling**

Writing Verilog

- Write a little bit of good code - debugging is hard so try to get it right on the first try
- Have a clear naming convention - is execute_pc the output of or input to execute?
- Reuse wires as much as reasonably possible - don't have immediate wires for each instruction variant
- Clearly separate stages - don't have intermixed code
- Recommended vscode extension

Stage contracts

- The hard part of pipelining is the communication between stages - not the stages themselves
- Without hazards, it's quite simple - input comes in, one cycle later output is ready
- But with hazards, things are no longer clear
- When the flush wire goes high for a cycle, does that immediately invalidate the output or does it take a cycle? When fetch receives a new PC, how many cycles till the instruction is ready? When a module has to stall, exactly which stall wires does it set and for how long?
- Treat stages as independent components - they only connect through wires defined in your contract

Debugging

- Don’t use print statements, you’ll get too much output
- DO NOT USE THE VSCODE WAVETRACE EXTENSION IT IS BUGGED (as of last year, and has not been updated since then so is still broken)
- GTKWave is vastly superior and is the only reasonable way to debug verilog
- X-forwarding works really well, just ssh with -X and launch gtkwave (use WSL on Windows 11)
- Add all the important wires for each stage, group them (press G), and optionally color code them
- Then make sure to save your layout, gtkwave will not remind you to save
- Searching for a value - select the wire you want to search, then search>pattern search 1>dropdown to “string”>plug in value>find next
- Right click on a signal, change the data format

Verilog Resources

- https://github.com/steveicarus/iverilog
- A Verilog Primer
  - Just note to use @(posedge clk) not @(*) in your code

Questions?