

Fixed -False

@ = True • = False

1.1)

underline = missed / wrong prediction

00.0 .00. 00.0 0.00 .0.0 0.00 $\frac{57}{118}$
accuracy

..00 .0.0 0... ...0 0.00 .00.0 00..

....0 0.00 .000 .00.

...0 0.00 .000 .00. 00.0 0.00 .00.

1.2)

static false first , option a (false first, then w/e happens next)

00.0 .00. 00.0 0.00 .0.0 0.00 $\frac{50}{118}$
accuracy

..00 .0.0 0... ...0 0.00 .00.0 00..

....0 0.00 .000 .00.

...0 0.00 .000 .00. 00.0 0.00 .00.

1.3)

Bimodal - False first, then w/e hasn't gone wrong twice in a row
assume we have been wrong once @ start

@@.@ .@@. @@.@ @.@@ .@.@@ @.@@ ...•• $\frac{71}{112}$

..@@ .@.@ @..^F ...@ @.@@ .@@. @@.. accuracy

F@ .@.@@ .@@@ .@@. ...^••

...@ @.@@ .@@@ .@@. @@.@ @.@@ .@@..

history

	P	C
..	0	1
•@	1	1
@•	0	1
@@	1	1

Prediction
Count of errors

2-Layer
Bi-modal

1.4)

..@@.@ .@@. @@.@ @.@@ .@.@ @.@@ ...••

assumed
.....

...@@ .@.@ @..• ...@ @.@@ .@@. @@..

....@ @.@@ .@@@ .@@.

...@ @.@@ .@@@ .@@. @.@@ @.@@ .@@.

$\frac{ab}{112}$
accuracy

Feb 27th

16-Bit Dynamic Encoding

Teeny AT

DATA Instructions

lod FB, [rA - 17] "register zero"

str [rC + 3], rZ

set rD, rA + 4

psh rA

pop rB

ALU Instructions

add rB, rA - 3

sub, mpy, div, mod, neg

and, or, xor, shif, rot

Control Instructions

Cmp

jmp

DJZ - Decrement + Jump if zero

DLX - Delay for a certain # of clock cycles

CAL - call

Registers

P.C. - Program Counter

S.P. - Stack Pointer

rZ - zero register

[rA - rE]

[r3 - r7]

faster execution
needs more power

Instruction Level Parallelism (ILP) "Pipelining"

Fetch

look-up

Decode

latches

(OF) Operand Fetch

flip-flop

Execute

cycles

(WB) Write Back

100 lines of code

500 cycles

104 cycles (using pipelining)

time

F D OF E WB

F D OF E WB

1

1

1

2

1

2

3

1

3

2

2

2 2

2

2 1

3 2

3 3

1

2

3

hazards (can make pipelining fail)

Structural → { set rA, 0xFE
add rA, r5
lod rB, [rA] two instructions
div rB, 4 need bus @ same time

Data →

Control → when one instruction relies on another instruction to finish before it can execute.

whenever we want to change what happens next, we can cause control hazards (jmp instructions are an example)

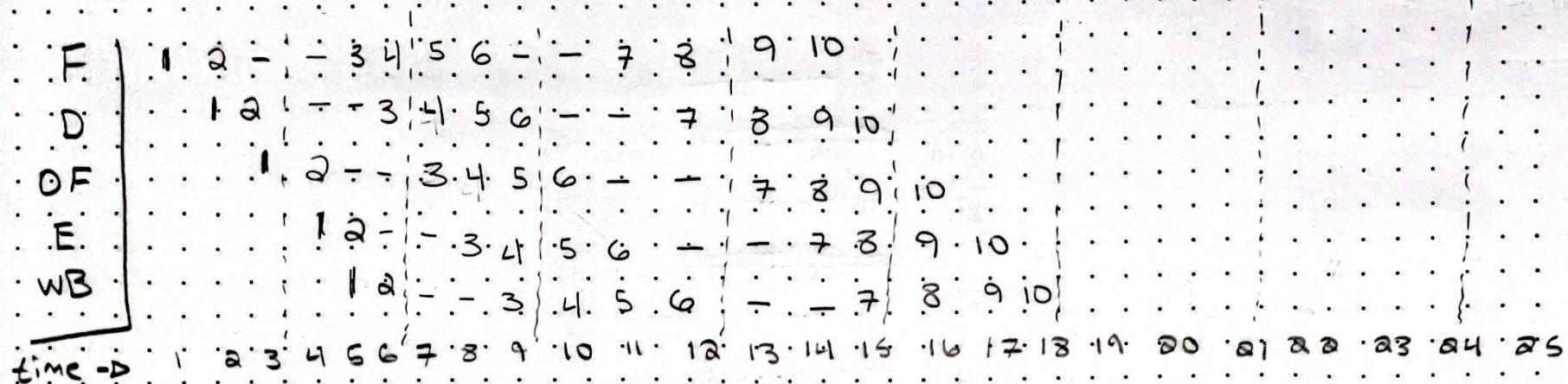
Feb 29 ±

$$\text{dist.} = (x_1 - x_0)^2 + (y_1 - y_0)^2 \quad \text{Euclidean sq.}$$

- ① lod rA, [x₁] ← loads on DF, Ex does nothing (bus)
- ② lod rB, [x₀]
- ③ sub rA, rB
- ④ mpy rA, rA
- ⑤ lod rB, [y₁]
- ⑥ lod rC, [y₀]
- ⑦ sub rB, rC
- ⑧ mpy rB, rB
- ⑨ add rA, rB
- ⑩ str [dist], rA

Pipeline Graph

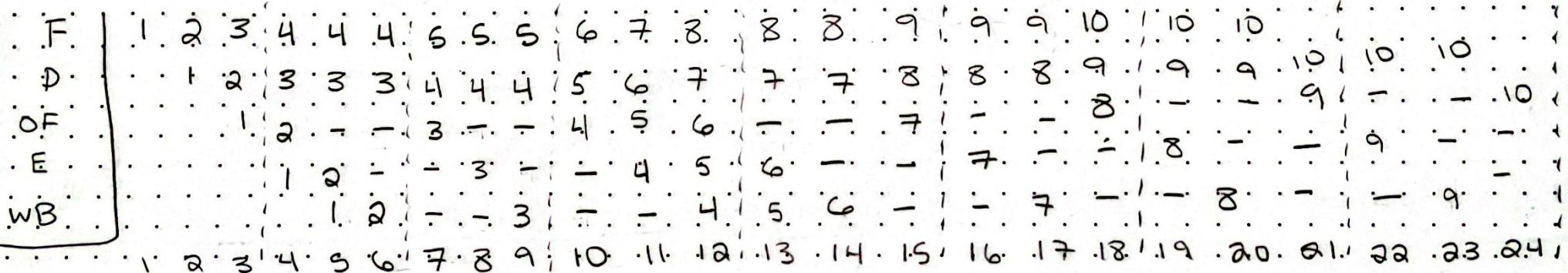
(structural)

No PipelinePipeline

(Data)

50

18



inst. rX, rY

dependency

inst. rZ, rX

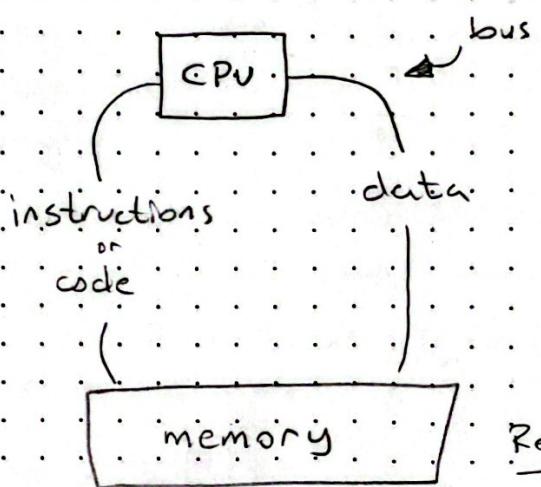
10 finishes

as expected

structural solutions

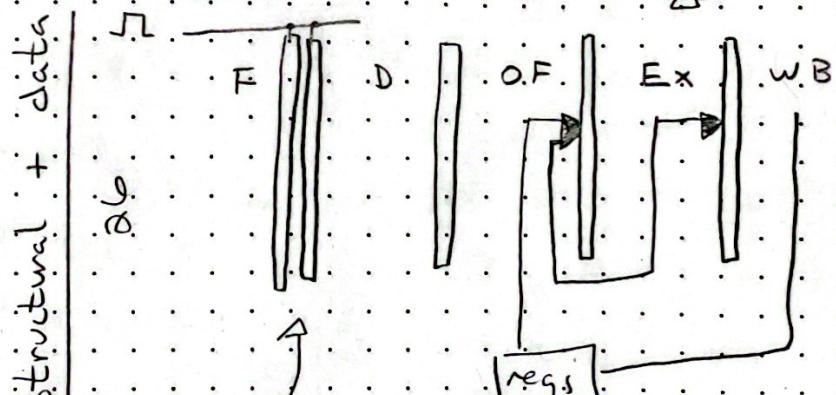
- wait one full cycle (stall)
- multiple buses

Harvard Architecture



look up.
rising edge
trigger
+
triggers

Register Forwarding



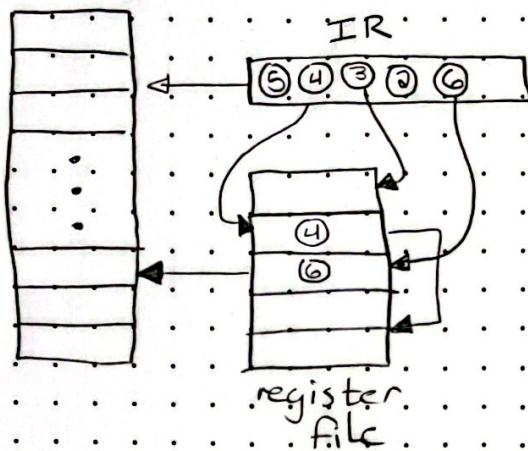
latch = temp storage device

	F	D	O	E	W	B
0	0	0	0	0	0	0
1	1	1	1	1	1	1
2	0	1	0	0	0	0
3	0	0	1	0	0	0
4	0	0	1	1	0	0
5	0	0	1	1	1	0
6	0	0	1	1	1	1
7	0	0	1	1	1	1
8	0	0	1	1	1	1
9	0	0	1	1	1	1
10	0	0	1	1	1	1
11	0	0	1	1	1	1
12	0	0	1	1	1	1
13	0	0	1	1	1	1
14	0	0	1	1	1	1
15	0	0	1	1	1	1
16	0	0	1	1	1	1
17	0	0	1	1	1	1
18	0	0	1	1	1	1
19	0	0	1	1	1	1
20	0	0	1	1	1	1
21	0	0	1	1	1	1
22	0	0	1	1	1	1
23	0	0	1	1	1	1
24	0	0	1	1	1	1
25	0	0	1	1	1	1
26	0	0	1	1	1	1
27	0	0	1	1	1	1
28	0	0	1	1	1	1
29	0	0	1	1	1	1
30	0	0	1	1	1	1
31	0	0	1	1	1	1
32	0	0	1	1	1	1
33	0	0	1	1	1	1
34	0	0	1	1	1	1
35	0	0	1	1	1	1
36	0	0	1	1	1	1
37	0	0	1	1	1	1
38	0	0	1	1	1	1
39	0	0	1	1	1	1
40	0	0	1	1	1	1
41	0	0	1	1	1	1
42	0	0	1	1	1	1
43	0	0	1	1	1	1
44	0	0	1	1	1	1
45	0	0	1	1	1	1
46	0	0	1	1	1	1
47	0	0	1	1	1	1
48	0	0	1	1	1	1
49	0	0	1	1	1	1
50	0	0	1	1	1	1
51	0	0	1	1	1	1
52	0	0	1	1	1	1
53	0	0	1	1	1	1
54	0	0	1	1	1	1
55	0	0	1	1	1	1
56	0	0	1	1	1	1
57	0	0	1	1	1	1
58	0	0	1	1	1	1
59	0	0	1	1	1	1
60	0	0	1	1	1	1
61	0	0	1	1	1	1
62	0	0	1	1	1	1
63	0	0	1	1	1	1
64	0	0	1	1	1	1
65	0	0	1	1	1	1
66	0	0	1	1	1	1
67	0	0	1	1	1	1
68	0	0	1	1	1	1
69	0	0	1	1	1	1
70	0	0	1	1	1	1
71	0	0	1	1	1	1
72	0	0	1	1	1	1
73	0	0	1	1	1	1
74	0	0	1	1	1	1
75	0	0	1	1	1	1
76	0	0	1	1	1	1
77	0	0	1	1	1	1
78	0	0	1	1	1	1
79	0	0	1	1	1	1
80	0	0	1	1	1	1
81	0	0	1	1	1	1
82	0	0	1	1	1	1
83	0	0	1	1	1	1
84	0	0	1	1	1	1
85	0	0	1	1	1	1
86	0	0	1	1	1	1
87	0	0	1	1	1	1
88	0	0	1	1	1	1
89	0	0	1	1	1	1
90	0	0	1	1	1	1
91	0	0	1	1	1	1
92	0	0	1	1	1	1
93	0	0	1	1	1	1
94	0	0	1	1	1	1
95	0	0	1	1	1	1
96	0	0	1	1	1	1
97	0	0	1	1	1	1
98	0	0	1	1	1	1
99	0	0	1	1	1	1
100	0	0	1	1	1	1
101	0	0	1	1	1	1
102	0	0	1	1	1	1
103	0	0	1	1	1	1
104	0	0	1	1	1	1
105	0	0	1	1	1	1
106	0	0	1	1	1	1
107	0	0	1	1	1	1
108	0	0	1	1	1	1
109	0	0	1	1	1	1
110	0	0	1	1	1	1
111	0	0	1	1	1	1
112	0	0	1	1	1	1
113	0	0	1	1	1	1
114	0	0	1	1	1	1
115	0	0	1	1	1	1
116	0	0	1	1	1	1
117	0	0	1	1	1	1
118	0	0	1	1	1	1
119	0	0	1	1	1	1
120	0	0	1	1	1	1
121	0	0	1	1	1	1
122	0	0	1	1	1	1
123	0	0	1	1	1	1
124	0	0	1	1	1	1
125	0	0	1	1	1	1
126	0	0	1	1	1	1
127	0	0	1	1	1	1
128	0	0	1	1	1	1
129	0	0	1	1	1	1
130	0	0	1	1	1	1
131	0	0	1	1	1	1
132	0	0	1	1	1	1
133	0	0	1	1	1	1
134	0	0	1	1	1	1
135	0	0	1	1	1	1
136	0	0	1	1	1	1
137	0	0	1	1	1	1
138	0	0	1	1	1	1
139	0	0	1	1	1	1
140	0	0	1	1	1	1
141	0	0	1	1	1	1
142	0	0	1	1	1	1
143	0	0	1	1	1	1
144	0	0	1	1	1	1
145	0	0	1	1	1	1
146	0	0	1	1	1	1
147	0	0	1	1	1	1
148	0	0	1	1	1	1
149	0	0	1	1	1	1
150	0	0	1	1	1	1
151	0	0	1	1	1	1
152	0	0	1	1	1	1
153	0	0	1	1	1	1
154	0	0	1	1	1	1
155	0	0	1	1	1	1
156	0	0	1	1	1	1
157	0	0	1	1	1	1
158	0	0	1	1	1	1
159	0	0	1	1	1	1
160	0	0	1	1	1	1
161	0	0	1	1	1	1
162	0	0	1	1	1	1
163	0	0	1	1	1	1
164	0	0	1	1	1	1
165	0	0	1	1	1	1
166	0	0	1	1	1	1
167	0	0	1	1	1	1
168	0	0	1	1	1	1
169	0	0	1	1	1	1
170	0	0	1	1	1	1
171	0	0	1	1	1	1
172	0	0	1	1	1	1
173	0	0	1	1	1	1
174	0	0	1	1	1	1
175	0	0	1	1	1	1
176	0	0	1	1	1	1
177	0	0	1	1	1	1
178	0	0	1	1	1	1
179	0	0	1	1	1	1
180	0	0	1	1	1	1
181	0	0	1	1	1	1
182	0	0	1	1	1	1
183	0	0	1	1	1	1
184	0	0	1	1	1	1
185	0	0	1	1	1	1
186	0	0	1	1	1	1
187	0	0	1	1	1	1
188	0	0	1	1	1	1
189	0	0	1	1	1	1
190	0	0	1	1	1	1
191	0	0	1	1	1	1
192	0	0	1	1	1	1
193	0	0	1	1	1	1
194	0	0	1	1	1	1
195	0	0	1	1	1	1
196	0	0	1	1	1	1
197	0	0	1	1	1	1
198	0	0	1	1	1	1
199	0	0	1	1	1	1
200	0	0	1	1	1	1
201	0	0	1	1	1	1
202	0	0	1	1	1	1
203	0	0	1	1	1	1
204	0	0	1	1	1	1
205	0	0	1	1	1	1
206	0	0	1	1	1	1
207	0	0	1	1	1	1
208	0	0	1	1	1	1
209	0	0	1	1	1	1
210	0	0	1	1	1	1
211	0	0	1	1	1	1
212	0	0	1	1	1	1
213	0	0	1	1	1	1
214	0	0	1	1	1	1
215	0	0	1	1	1	1
216	0	0	1	1	1	1
217	0	0	1	1</		

Mar 5 -

addressing (modes)

Memory



- ① Implicit addressing (don't tell it what to do)
alpha 'x'
- ② Immediate addressing (some value)
- ③ Register addressing (accessing directly)
- ④ Register by register addressing (register points to another register)
- ⑤ Indirect addressing (immediate value)
- ⑥ Register indirection addressing

control hazards

```
for(int i=3; i < 5; i++) {  
    if(i%2==0){  
        i++;  
    }  
}
```

Asm

```
set rA, 3  
!top  
2. cmp rA, 5  
3. jge !done  
4. set rB, rA  
5. mod rB, 2 // EQ: 1 if equal  
6. jne !continue  
7. inc rA  
  
!continue  
8. inc rA  
9. jmp !top
```

Pipeline on next
page

control + data hazards

* = flush

F	1. 2 3 3 4 4 5 6 6 7 7 8	*	8 9	*	2 3 4 4 5 6 6 7 7	
D	1 3 2 3 3 4 5 5 6 6 7 - - 8 9	-	2 3 3 4	5 5 6	6 7	
OF	1 - 2 - 3 4 - 5 - 6 - - 8 9 - 2 - 3 4 - 5 - 6 7					
E	1 - 2 - 3 4 - 5 - 6 - - 8 9 - 2 - 3 4 - 5 - 6					
WB	1 - 2 - 3 4 - 5 - 6 - - 8 9 - 2 - 3 4 - 5 -					
	1. 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7					

- use register forwarding
 - unconditional JMP known at decode, so flush
 - conditional JMP known at execute ; so flush (OF grabs address, Eq flag + gt flag)

Mar 7th

Hand-out

Set rB, 5

mpy rA, rB

jmp !top

// more code

helps pipeline

Set rB, 5

jmp !top

mpy rA, rB

study

-caching

(put 'true' cases at top of if statement to make code more efficient)

Branch Prediction

- ① Fixed False - We assume all conditional branches will not branch (are false)
↳ easiest to predict / cheapest

- ② Static False First

a) jmp \Rightarrow true

je/jne \Rightarrow false first, after, whatever happens first

↳ have to use caching, storing info about how instructions should work

b) false first; after; whatever was first by address.

- ③ Bimodal - False first; whatever hasn't gone wrong twice in a row next
↳ keep predicting what we are predicting until we are wrong twice in a row

- ④ Two-Layer Bimodal -

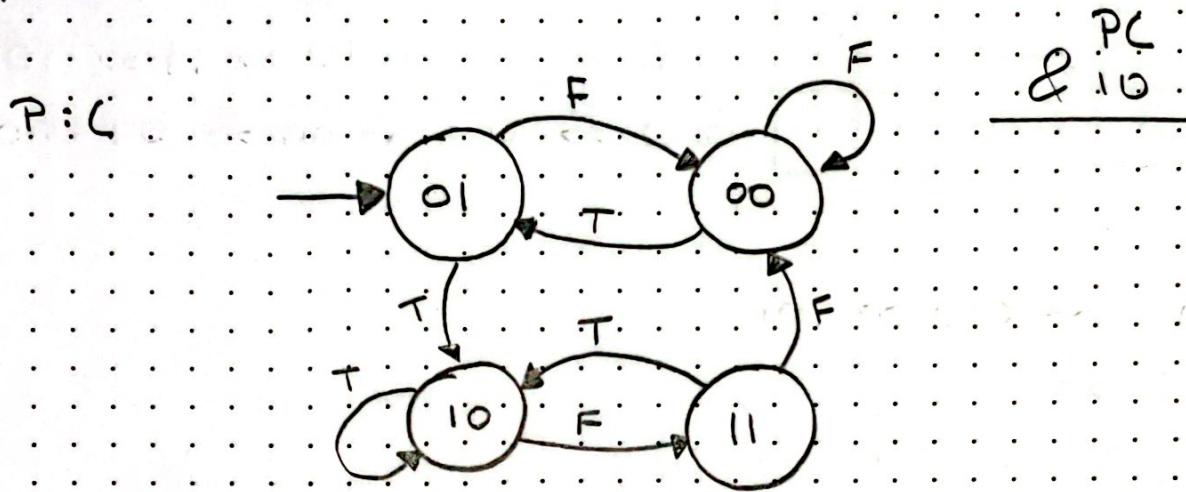
diff op codes

work-sheet

2 bits need to be added to use Bimodal

cnt = how many wrongs

prediction = T/F



Mar 14th

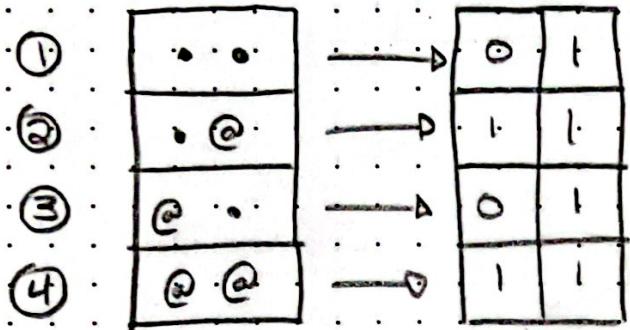
Line 13 of HW 1 has operands reversed

[Intel PIN demonstration]

- Ran Bubble Sort code to show how he used PIN to show branching patterns
- used g++ to compile, then logged branches of compiler

Two-layer Bi-modal

Assume 0 branches were not taken (specific to HW3, not all 2-layer Bi-modal)



10 >> 11 >> 00 >> 01 >> 00 *10 >> 11 >> 10 >> 11 >> 10 *10 >> 11 >> 10 >> 11 >> 10 *00 >> 01 >> 00 >> 01 >> 00

① * >> 01 >> 00 >> 01

② * >> 11 >> 10

③ * >> 11 >> 10 >> 11 >> 10

④