Basic Pipelined Processor

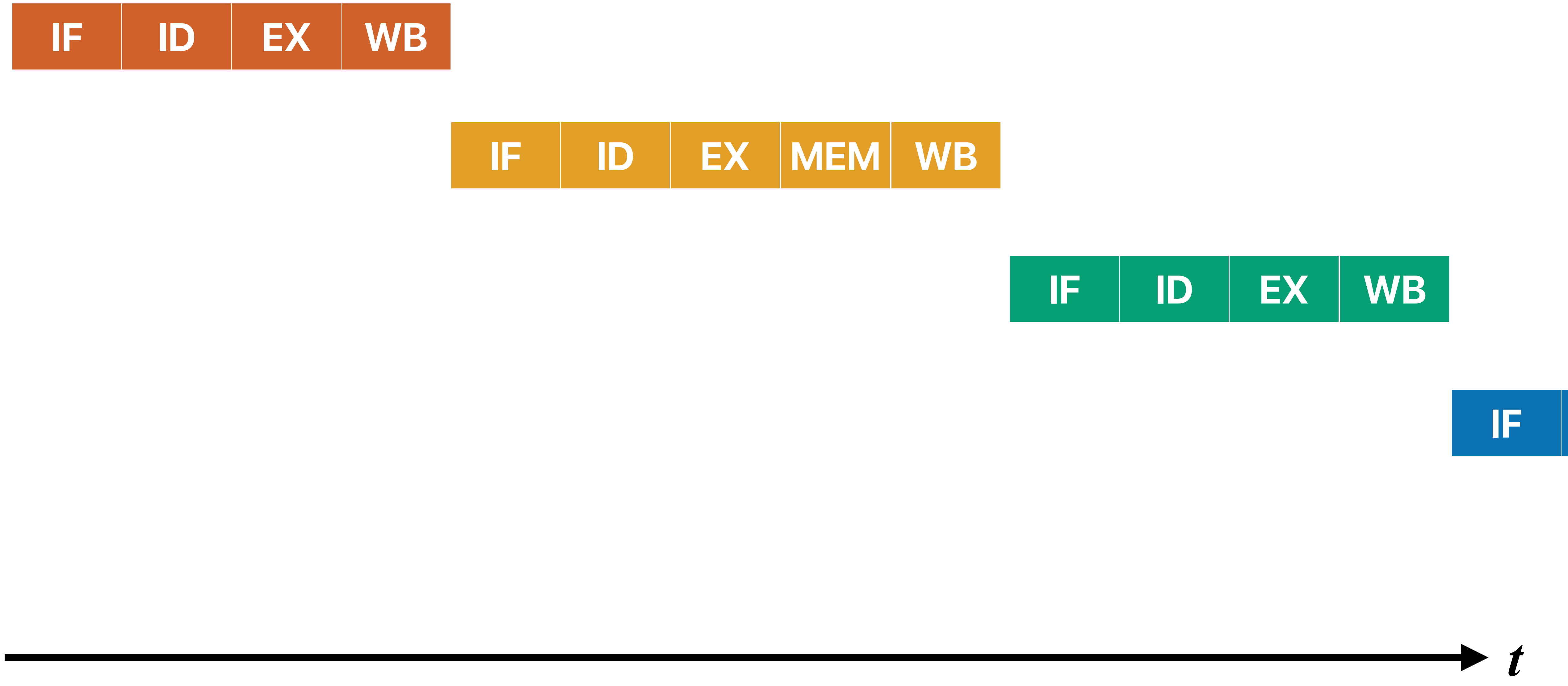
Hung-Wei Tseng

Tasks in RISC-V ISA

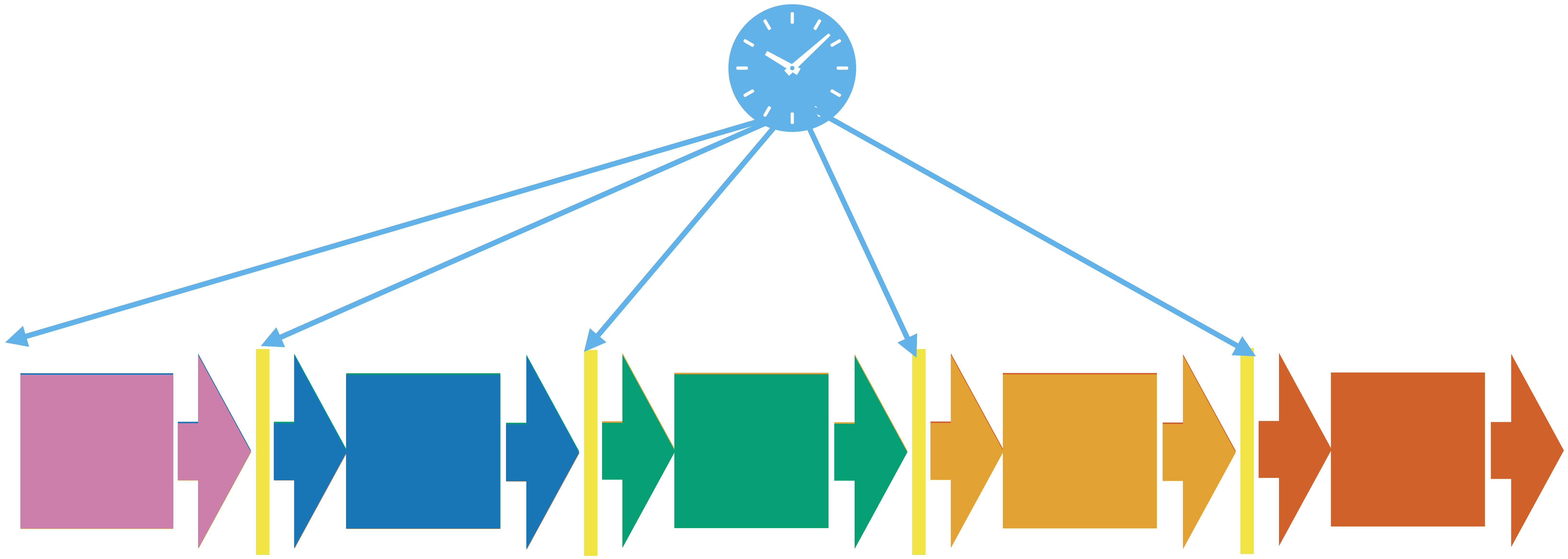
- Instruction Fetch (**IF**) — fetch the instruction from memory
- Instruction Decode (**ID**)
  - Decode the instruction for the desired operation and operands
  - Reading source register values
- Execution (**EX**)
  - ALU instructions: Perform ALU operations
  - Conditional Branch: Determine the branch outcome (taken/not taken)
  - Memory instructions: Determine the effective address for data memory access
- Data Memory Access (**MEM**) — Read/write memory
- Write Back (**WB**) — Present ALU result/read value in the target register
- Update PC
  - If the branch is taken — set to the branch target address
  - Otherwise — advance to the next instruction — current PC + 4

Simple implementation w/o branch

```
add x1, x2, x3
ld  x4, 0(x5)
sub x6, x7, x8
sub x9,x10,x11
sd  x1, 0(x12)
```

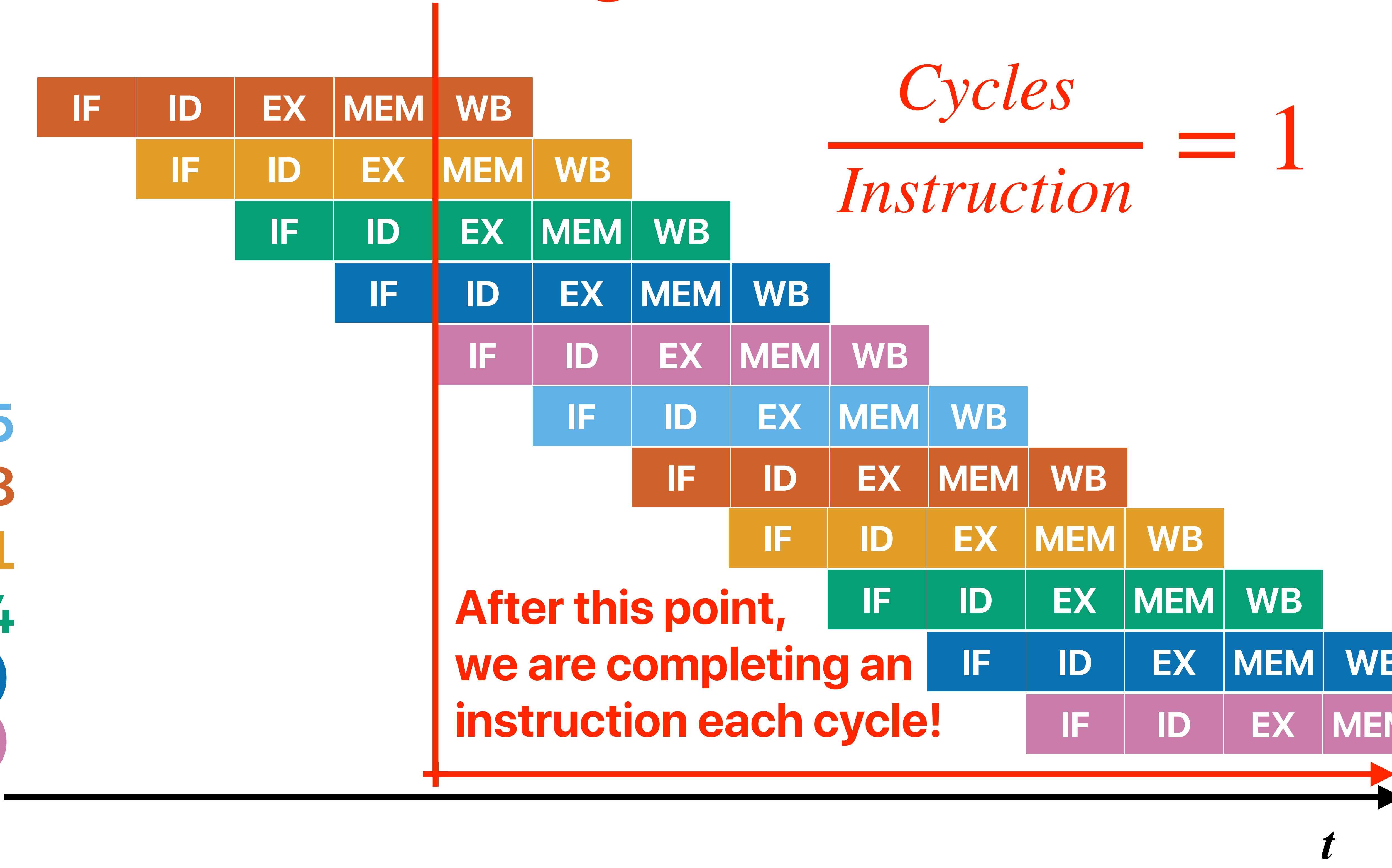


Pipelining



Pipelining

```
add x1,  x2,  x3
ld  x4,  0(x5)
sub x6,  x7,  x8
sub x9,x10,x11
sd  x1,  0(x12)
xor x13,x14,x15
and x16,x17,x18
add x19,x20,x21
sub x22,x23,x24
ld  x25, 4(x26)
sd  x27, 0(x28)
```


IF
ID
EX
MEM
WB
After this point, we are completing an instruction each cycle!
t

$$\frac{\textit{Cycles}}{\textit{Instruction}} = 1$$

Which version is faster?

A

```
for(i=0;i<1000000000;i++)
{
      sum+=data[(i*15) & 131071];
}
```

B

```
for(i=0;i<1000000000;i++)
{
      sum+=data[((i << 4) - i) & 131071];
}
```

- Both version A and B produces the same output. Without compiler optimization, which version of code would have better performance?
  A. Version A
  B. Version B
  [✓ C. They are about the same (less than 5% difference)]

Outline

- Pipeline Hazards
- Structural Hazards
- Control Hazards
- Dynamic Branch Predictions

Poll close in 1:30

Can we get them right?

- Given a simple pipelined RISC-V processor that we discussed so far, how many of the following code snippets can be executed with expected outcome?

	I	II	III	IV
a	`add x1, x2, x3`	`add x1, x2, x3`	`add x1, x2, x3`	`add x1, x2, x3`
b	`ld x4, 0(`**`x1`**`)`	`ld x4, 0(x5)`	`ld x4, 0(x5)`	`ld x4, 0(x5)`
c	`sub x6, x7, x8`	`sub x6, x7, x8`	**`bne x0, x7, L`**	`sub x6, x7, x8`
d	`sub x9,x10,x11`	`sub x9, `**`x1`**`, x10`	`sub x9,x10,x11`	`sub x9,x10,x11`
e	`sd x1, 0(x12)`	`sd x11, 0(x12)`	`sd x1, 0(x12)`	`sd x1, 0(x12)`

A. 0
B. 1
C. 2
D. 3
E. 4

Poll close in 1:30

Can we get them right?

1 — Instinct

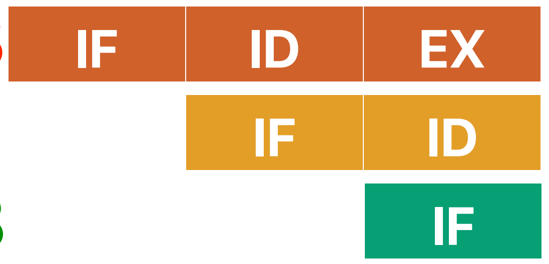
- Given a simple pipelined RISC-V processor that we discussed so far, how many of the following code snippets can be executed with expected outcome?

	I	II	III	IV
a	`add x1, x2, x3`	`add x1, x2, x3`	`add x1, x2, x3`	`add x1, x2, x3`
b	`ld x4, 0(`**`x1`**`)`	`ld x4, 0(x5)`	`ld x4, 0(x5)`	`ld x4, 0(x5)`
c	`sub x6, x7, x8`	`sub x6, x7, x8`	**`bne x0, x7, L`**	`sub x6, x7, x8`
d	`sub x9,x10,x11`	`sub x9, `**`x1`**`, x10`	`sub x9,x10,x11`	`sub x9,x10,x11`
e	`sd x1, 0(x12)`	`sd x11, 0(x12)`	`sd x1, 0(x12)`	`sd x1, 0(x12)`

A. 0
B. 1
C. 2
D. 3
E. 4

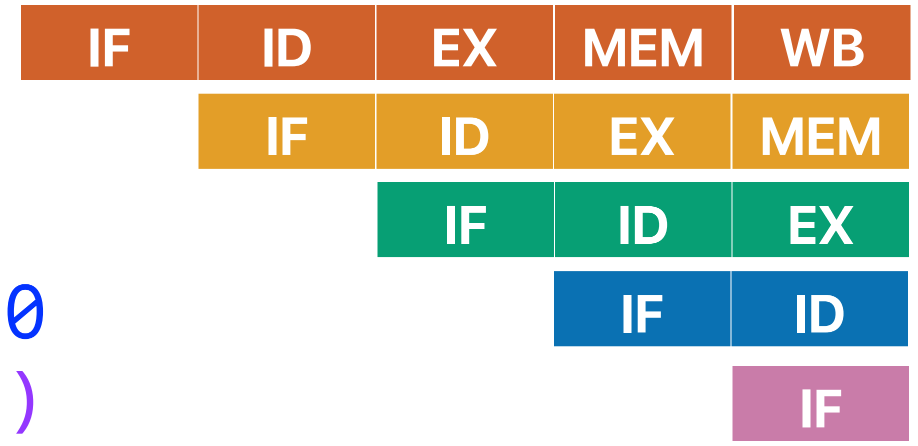
Draw the pipeline diagrams

```
add x1, x2, x3
ld  x4, 0(x1)
sub x6, x7, x8
sub x9,x10,x11
sd  x1, 0(x12)
```



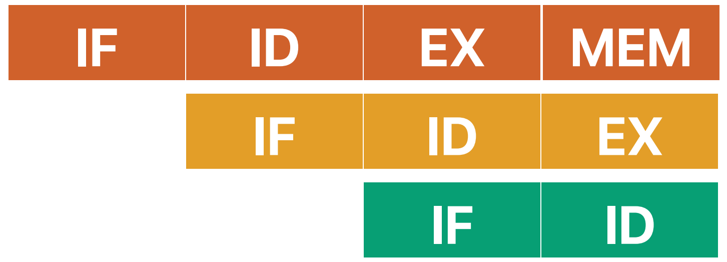
The desired value of x1 is not ready yet

```
add x1, x2, x3
ld  x4, 0(x5)
sub x6, x7, x8
sub x9, x1, x10
sd  x11, 0(x12)
```



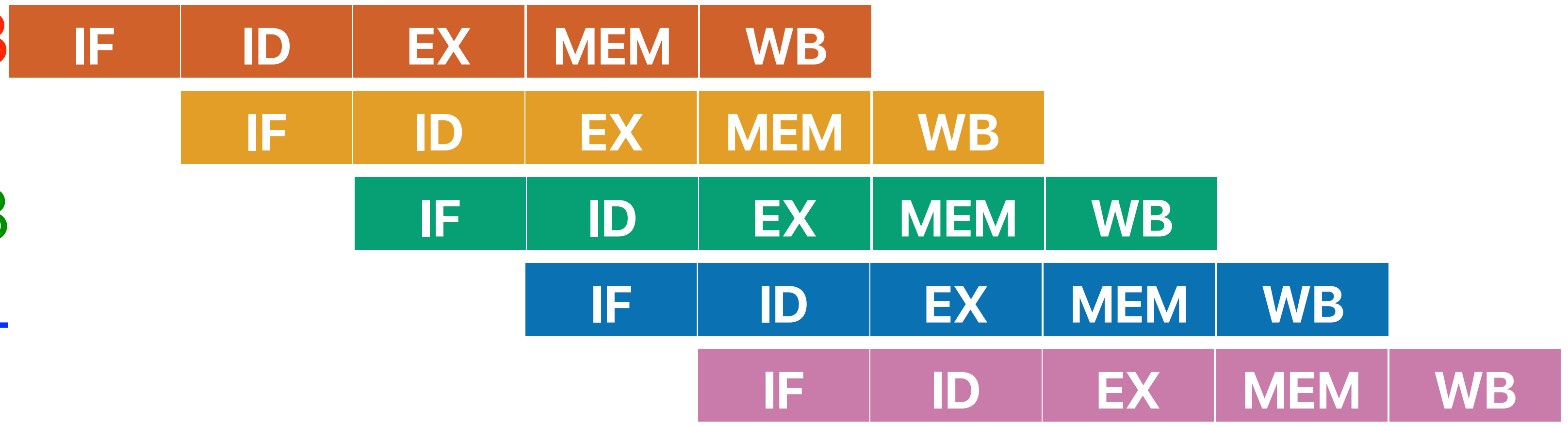
Both instructions need x1

```
add x1, x2, x3
ld  x4, 0(x5)
bne x0, x7, L
sub x9,x10,x11
sd  x1, 0(x12)
```



Doesn't know what to fetch at this moment

```
add x1, x2, x3
ld  x4, 0(x5)
sub x6, x7, x8
sub x9,x10,x11
sd  x1, 0(x12)
```



Can we get them right?

- Given a simple pipelined RISC-V processor that we discussed so far, how many of the following code snippets can be executed with expected outcome?

	I	II	III	IV
a	`add x1, x2, x3`	`add x1, x2, x3`	`add x1, x2, x3`	`add x1, x2, x3`
b	`ld x4, 0(`**`x1`**`)`	`ld x4, 0(x5)`	`ld x4, 0(x5)`	`ld x4, 0(x5)`
c	`sub x6, x7, x8`	`sub x6, x7, x8`	**`bne x0, x7, L`**	`sub x6, x7, x8`
d	`sub x9,x10,x11`	`sub x9, `**`x1`**`, x10`	`sub x9,x10,x11`	`sub x9,x10,x11`
e	`sd x1, 0(x12)`	`sd x11, 0(x12)`	`sd x1, 0(x12)`	`sd x1, 0(x12)`

A. 0
[✓ B. 1]
C. 2
D. 3
E. 4

b cannot get x1 produced by a before WB

both a and d are accessing x1 at the 5th cycle

We don't know if d & e will be executed or not until c finishes

Pipeline hazards

Three pipeline hazards

- Structural hazards — resource conflicts cannot support simultaneous execution of instructions in the pipeline
- Control hazards — the PC can be changed by an instruction in the pipeline
- Data hazards — an instruction depending on a the result that's not yet generated or propagated when the instruction needs that

Can we get them right?

- Given a simple pipelined RISC-V processor that we discussed so far, how many of the following code snippets can be executed with expected outcome?

	I	II	III	IV
a	`add x1, x2, x3`	`add x1, x2, x3`	`add x1, x2, x3`	`add x1, x2, x3`
b	`ld x4, 0(`**`x1`**`)`	`ld x4, 0(x5)`	`ld x4, 0(x5)`	`ld x4, 0(x5)`
c	`sub x6, x7, x8`	`sub x6, x7, x8`	**`bne x0, x7, L`**	`sub x6, x7, x8`
d	`sub x9,x10,x11`	`sub x9, `**`x1`**`, x10`	`sub x9,x10,x11`	`sub x9,x10,x11`
e	`sd x1, 0(x12)`	`sd x11, 0(x12)`	`sd x1, 0(x12)`	`sd x1, 0(x12)`

A. 0
[✓ B. 1]
C. 2
D. 3
E. 4

b cannot get x1 produced by a before WB

Data Hazard

both a and d are accessing x1 at the 5th cycle

Structural Hazard

We don't know if d & e will be executed or not until c finishes

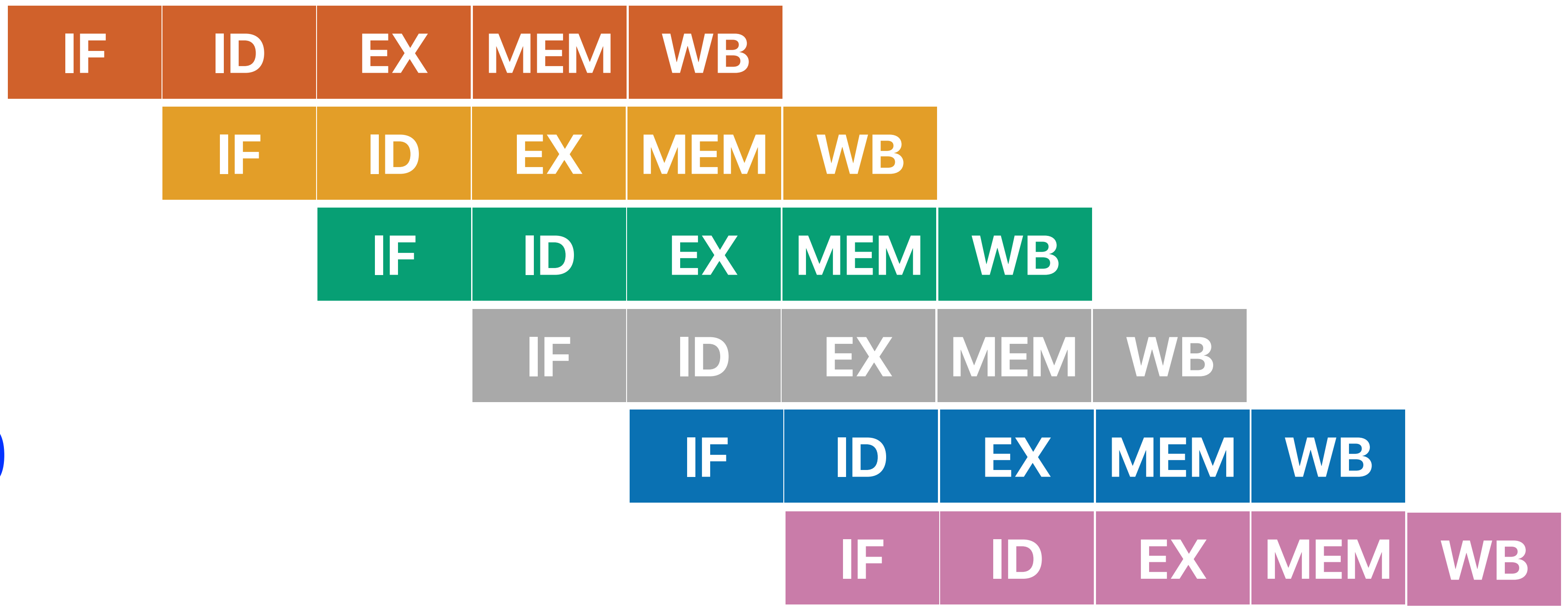
Control Hazard

Structural Hazards

Dealing with the conflicts between ID/WB

- The same register cannot be read/written at the same cycle
- Solution: insert no-ops (e.g, `add x0,x0,x0`) between them
- Drawback
  - If the number of pipeline stages changes, the code won't work
  - Slow

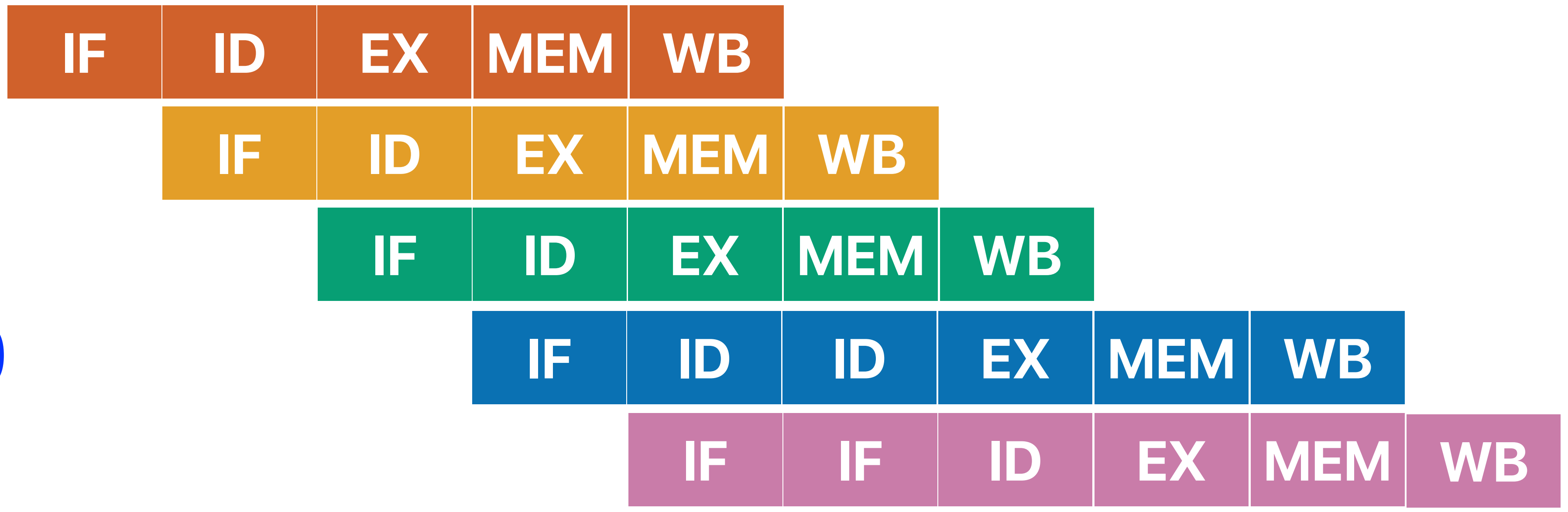
```
add x1, x2, x3
ld  x4, 0(x5)
sub x6, x7, x8
add x0, x0, x0
sub x9, x1, x10
sd  x11, 0(x12)
```



Dealing with the conflicts between ID/WB

- The same register cannot be read/written at the same cycle
- Solution: stall the later instruction, allowing the write to present the change in the register and the later can get the desired value
- Drawback: slow

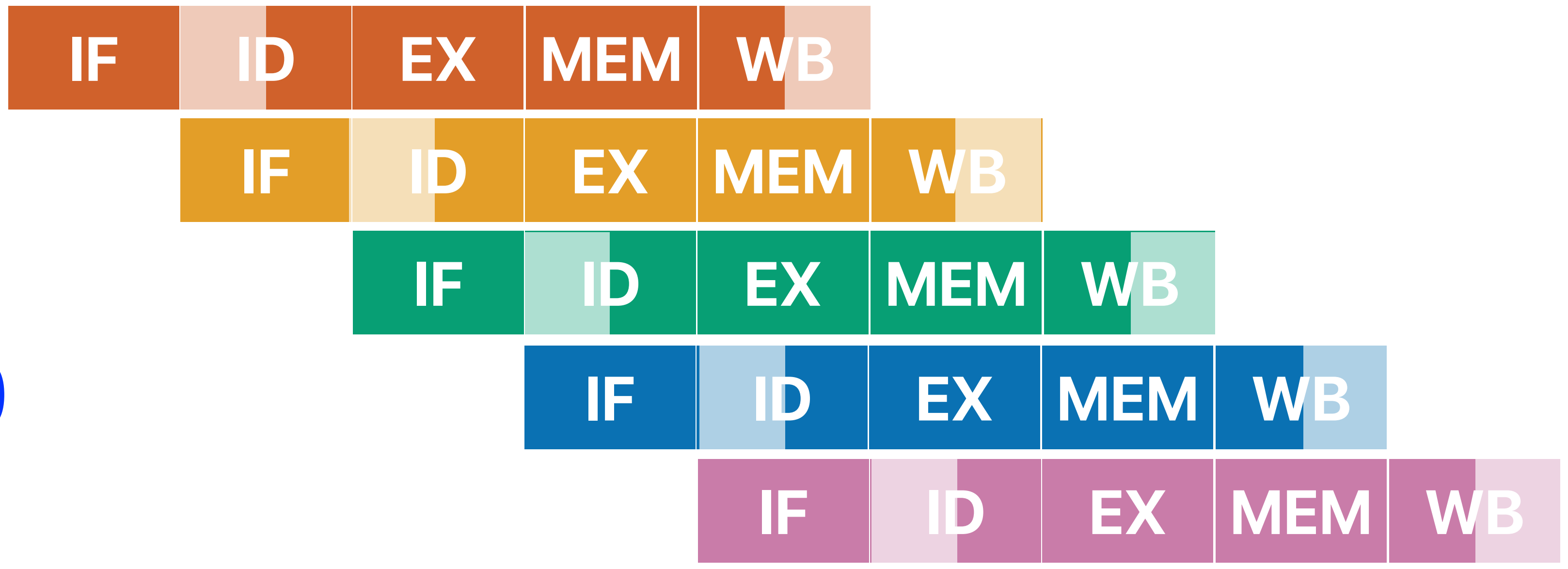
```
add x1, x2, x3
ld  x4, 0(x5)
sub x6, x7, x8
sub x9, x1, x10
sd  x11, 0(x12)
```



Dealing with the conflicts between ID/WB

- The same register cannot be read/written at the same cycle
- Better solution: write early, read late
  - Writes occur at the clock edge and complete long enough before the end of the clock cycle.
  - This leaves enough time for outputs to settle for reads
  - The revised register file is the default one from now!

```
add x1, x2, x3
ld  x4, 0(x5)
sub x6, x7, x8
sub x9, x1, x10
sd  x11, 0(x12)
```



Poll close in 1:30

Structural Hazards

- What pair of instructions will be problematic if we allow ALU instructions to skip the “MEM” stage?

```
a: ld  x1, 0(x2)
b: add x3, x4, x5
c: sub x6, x7, x8
d: sub x9,x10,x11
e: sd  x1, 0(x12)
```

A. a & b

B. a & c

C. b & e

D. c & e

E. None

Poll close in 1:30

1 — Instinct

Structural Hazards

- What pair of instructions will be problematic if we allow ALU instructions to skip the "MEM" stage?

```
a: ld  x1, 0(x2)
b: add x3, x4, x5
c: sub x6, x7, x8
d: sub x9,x10,x11
e: sd  x1, 0(x12)
```

A. a & b

B. a & c

C. b & e

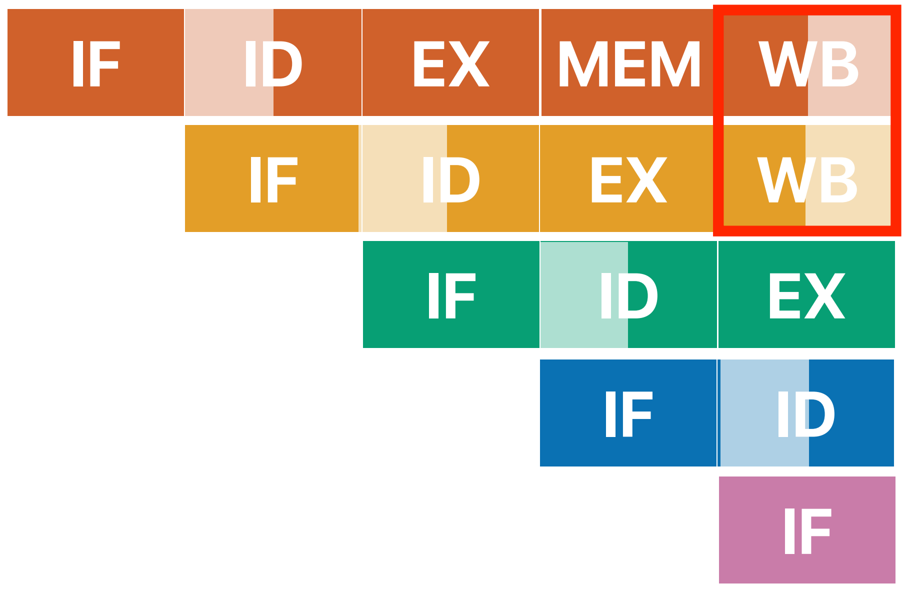
D. c & e

E. None

Structural Hazards

- What pair of instructions will be problematic if we allow ALU instructions to skip the “MEM” stage?

```
a: ld  x1, 0(x2)
b: add x3, x4, x5
c: sub x6, x7, x8
d: sub x9,x10,x11
e: sd  x1, 0(x12)
```



A. [✓ a & b]

B. a & c

C. b & e

D. c & e

E. None

Structural Hazards

- Stall can address the issue — but slow
- Improve the pipeline unit design to allow parallel execution

Control Hazards

Poll close in 1:30

The impact of control hazards

- Assuming that we have an application with 20% of branch instructions and the instruction stream incurs no data hazards. When there is a branch, we disable the instruction fetch and insert no-ops until we can determine the PC. What's the average CPI if we execute this program on the 5-stage RISC-V pipeline?
  A. 1
  B. 1.2
  C. 1.4
  D. 1.6
  E. 1.8

Poll close in 1:30

The impact of control hazards

1 — Instinct

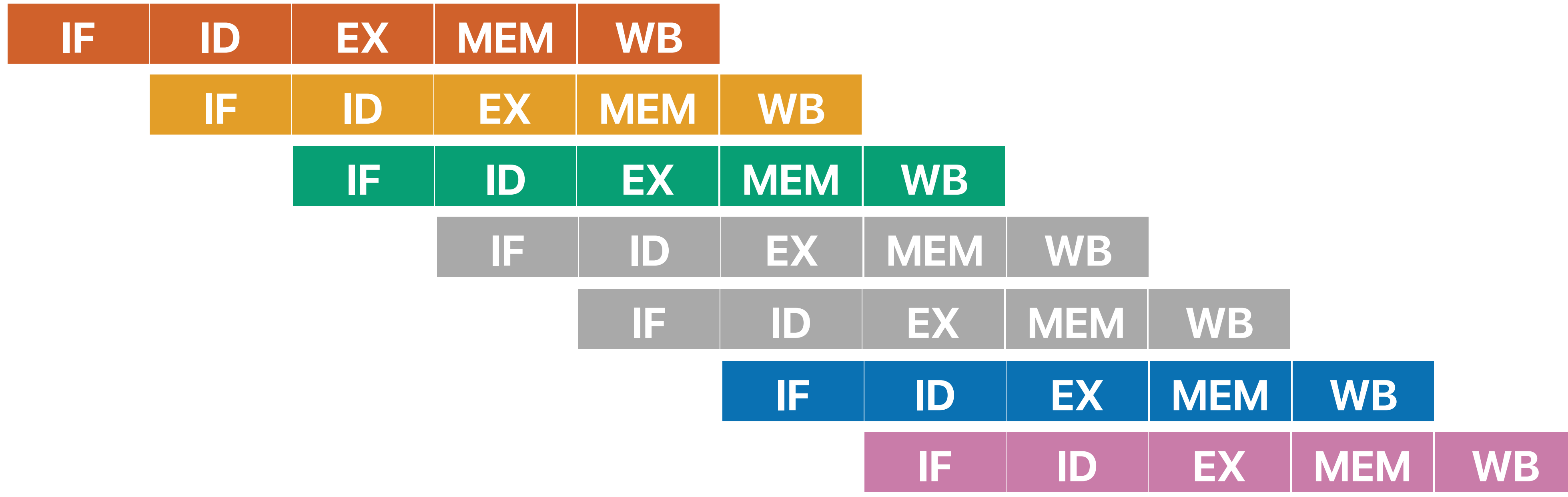
- Assuming that we have an application with 20% of branch instructions and the instruction stream incurs no data hazards. When there is a branch, we disable the instruction fetch and insert no-ops until we can determine the PC. What's the average CPI if we execute this program on the 5-stage RISC-V pipeline?
  A. 1
  B. 1.2
  C. 1.4
  D. 1.6
  E. 1.8

The impact of control hazards

- Assuming that we have an application with 20% of branch instructions and the instruction stream incurs no data hazards. When there is a branch, we disable the instruction fetch and insert no-ops until we can determine the PC. What's the average CPI if we execute this program on the 5-stage RISC-V pipeline?

  A. 1
  B. 1.2
  [✓ C. 1.4]
  D. 1.6
  E. 1.8

```
add x1, x2, x3
ld  x4, 0(x5)
bne x0, x7, L
add x0, x0, x0
add x0, x0, x0
sub x9,x10,x11
sd  x1, 0(x12)
```



$$1 + 20\% \times 2 = 1.4$$

Poll close in 1:30

Why can't we proceed without stalls/no-ops?

- How many of the following statements are true regarding why we have to stall for each branch in the current pipeline processor
  ① The target address when branch is taken is not available for instruction fetch stage of the next cycle
  ② The target address when branch is not-taken is not available for instruction fetch stage of the next cycle
  ③ The branch outcome cannot be decided until the comparison result of ALU is not out
  ④ The next instruction needs the branch instruction to write back its result
  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

Poll close in 1:30

Why can’t we proceed without stalls/no-

1 — Instinct

- How many of the following statements are true regarding why we have to stall for each branch in the current pipeline processor
  - ① The target address when branch is taken is not available for instruction fetch stage of the next cycle
  - ② The target address when branch is not-taken is not available for instruction fetch stage of the next cycle
  - ③ The branch outcome cannot be decided until the comparison result of ALU is not out
  - ④ The next instruction needs the branch instruction to write back its result
  - A. 0
  - B. 1
  - C. 2
  - D. 3
  - E. 4

Why can't we proceed without stalls/no-ops?

- How many of the following statements are true regarding why we have to stall for each branch in the current pipeline processor
  - ① ✓ The target address when branch is taken is not available for instruction fetch stage of the next cycle
  - ② The target address when branch is not-taken is not available for instruction fetch stage of the next cycle
  - ③ ✓ The branch outcome cannot be decided until the comparison result of ALU is not out
  - ④ The next instruction needs the branch instruction to write back its result
  - A. 0
  - B. 1
  - [✓ C. 2]
  - D. 3
  - E. 4

Dynamic Branch Prediction

Why can't we proceed without stalls/no-ops?

- How many of the following statements are true regarding why we have to stall for each branch in the current pipeline processor
  - ✓ ① The target address when branch is taken is not available for instruction fetch stage of the next cycle **You need a cheatsheet for that — branch target buffer**
  - ② The target address when branch is not-taken is not available for instruction fetch stage of the next cycle **You need to predict that — history/states**
  - ✓ ③ The branch outcome cannot be decided until the comparison result of ALU is not out
  - ④ The next instruction needs the branch instruction to write back its result

  A. 0

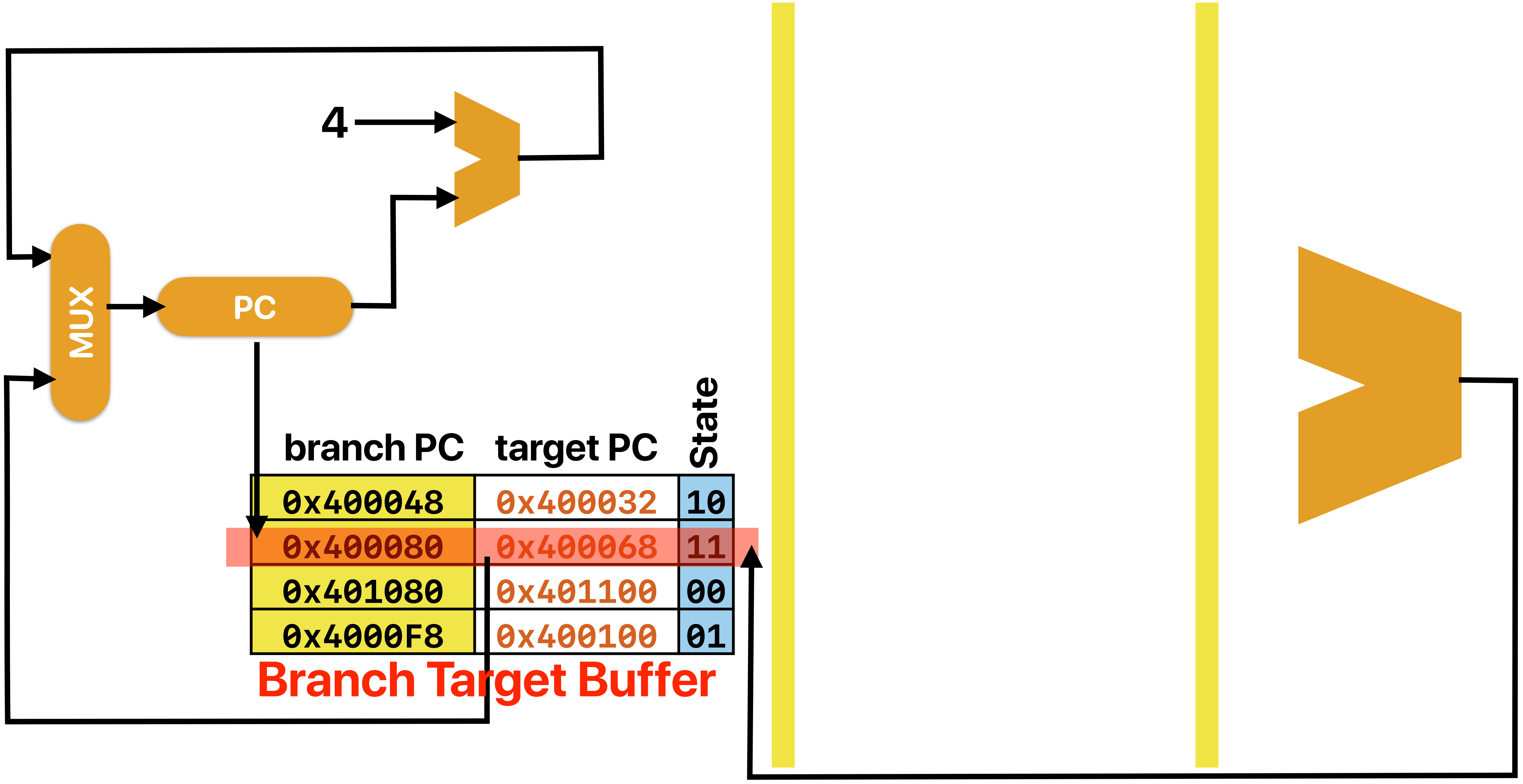
  B. 1

  [✓ C. 2]

  D. 3

  E. 4

A basic dynamic branch predictor



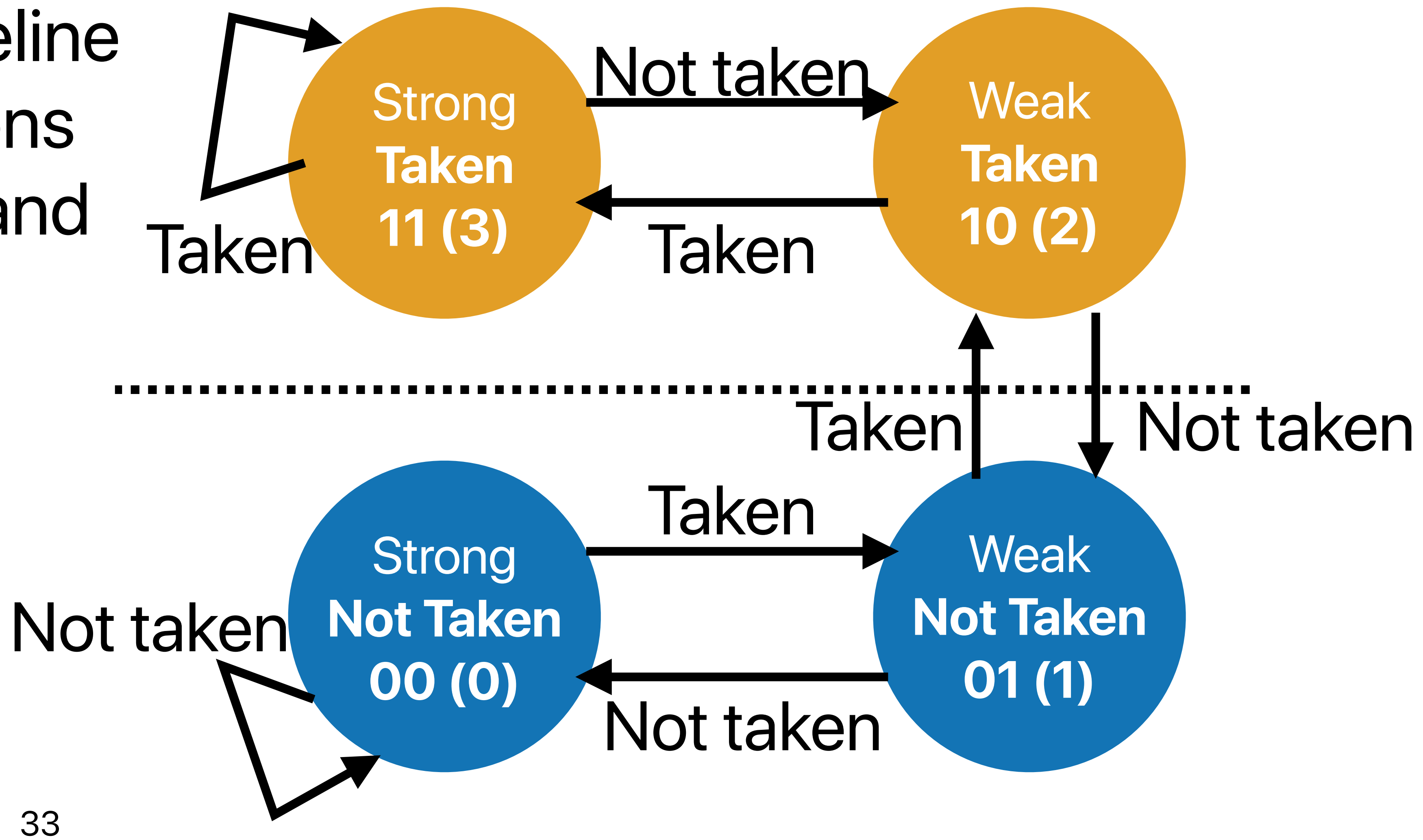
branch PC	target PC	State
0x400048	0x400032	10
0x400080	0x400068	11
0x401080	0x401100	00
0x4000F8	0x400100	01

2-bit/Bimodal local predictor

- Local predictor — every branch instruction has its own state
- 2-bit — each state is described using 2 bits
- Change the state based on **actual** outcome
- If we guess right — no penalty
- If we guess wrong — flush (clear pipeline registers) for mis-predicted instructions that are currently in IF and ID stages and reset the PC

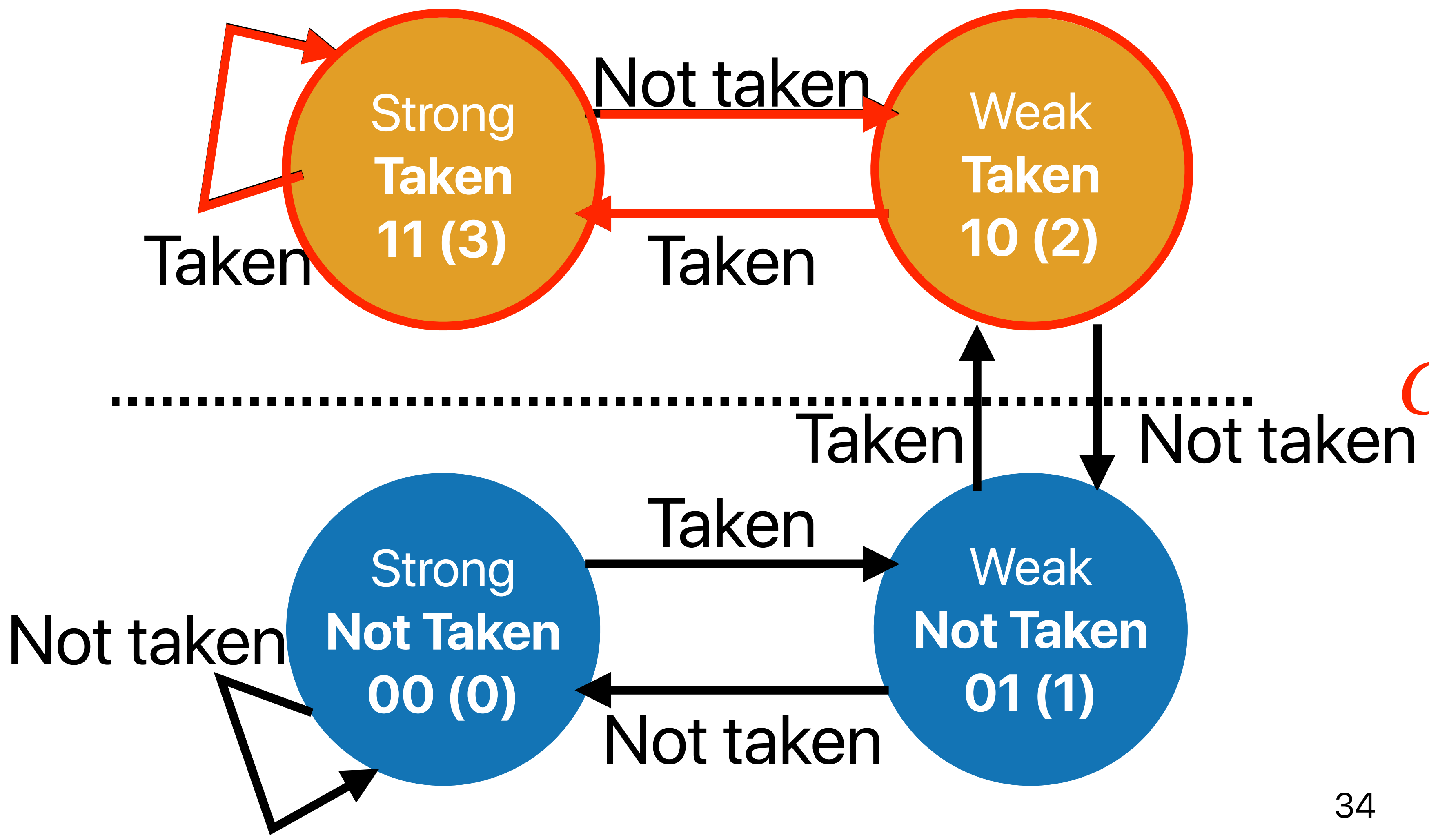
branch PC	target PC	State
0x400048	0x400032	10
0x400080	0x400068	11
0x401080	0x401100	00
0x4000F8	0x400100	01

Predict Taken



2-bit local predictor

```
i = 0;
do {
     sum += a[i];
} while(++i < 10);
```



i	state	predict	actual
1	10	T	T
2	11	T	T
3	11	T	T
4-9	11	T	T
10	11	T	NT

90% accuracy!

$$CPI_{average} = 1 + 20\% \times 10\% \times 2 = 1.04$$

Poll close in 1:30

2-bit local predictor

- What's the overall branch prediction (include both branches) accuracy for this nested for loop?

```
i = 0;
do {
    if( i % 2 != 0) // Branch X, taken if i % 2 == 0
       a[i] *= 2;
    a[i] += i;
} while ( ++i < 100)// Branch Y
```

(assume all states started with 00)

A. ~25%
B. ~33%
C. ~50%
D. ~67%
E. ~75%

Poll close in 1:30

1 — Instinct

2-bit local predictor

- What's the overall branch prediction (include both branches) accuracy for this nested for loop?

```
i = 0;
do {
    if( i % 2 != 0) // Branch X, taken if i % 2 == 0
       a[i] *= 2;
    a[i] += i;
} while ( ++i < 100)// Branch Y
```

(assume all states started with 00)

A. ~25%
B. ~33%
C. ~50%
D. ~67%
E. ~75%

2-bit local predictor

- What's the overall branch prediction (include both branches) accuracy for this nested for loop?

```
i = 0;
do {
    if( i % 2 != 0) // Branch X, taken if i % 2 == 0
        a[i] *= 2;
    a[i] += i;
} while ( ++i < 100)// Branch Y
```

Can we do a better job?

(assume all states started with 00)

A. ~25%
B. ~33%
C. ~50%
D. ~67%
[✓ E. ~75%]

i	branch?	state	prediction	actual
0	X	00	NT	T
1	Y	00	NT	T
1	X	01	NT	NT
2	Y	01	NT	T
2	X	00	NT	T
3	Y	10	T	T
3	X	01	NT	NT
4	Y	11	T	T
4	X	00	NT	T
5	Y	11	T	T
5	X	01	NT	NT
6	Y	11	T	T
6	X	00	NT	T
7	Y	11	T	T

For branch Y, almost 100%,
For branch X, only 50%

Midterm Logistics

For midterm

- Release Tuesday 0:00am, turn in before **Friday 11:59:00pm**
- You can only open it once and you have to finish a total of 30 questions within 80 minutes.
- You may open book, but you have to bare the risks of not being able to finish them
- No cheating is allowed
  - Discussion on Piazza/online forums, with any other person is considered as cheating — we already allow you to open book
  - We have free answer questions — those should not be identical

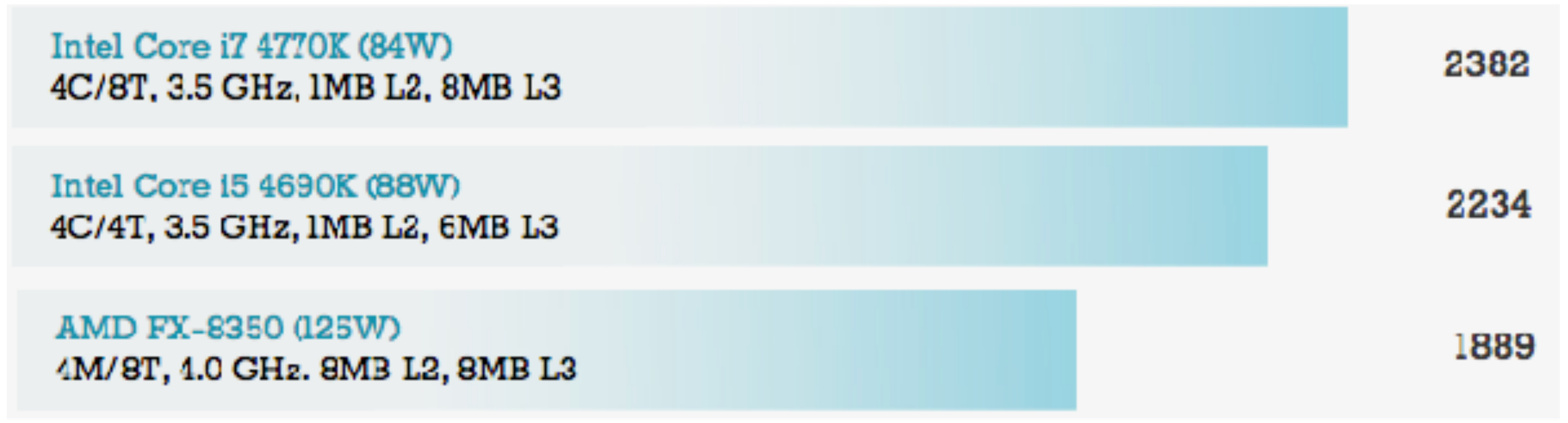
Format of the midterm

- Multiple choices * 20 — like your poll/reading quizzes multiple choices questions
- Homework style free-answer questions * 10
  - You need to clearly write down the original form of the applied equation/formula
  - You need to replace each term accordingly with numbers
  - You will have some credits for right equations even though the final number isn’t correct
  - You will receive 0 credits if we only see the numbers

Sample Midterm

Identify the performance bottleneck

- Why does an Intel Core i7 @ 3.5 GHz usually perform better than an Intel Core i5 @ 3.5 GHz or AMD FX-8350@4GHz?



Sysbench 2014 from http://www.anandtech.com/

A. Because the instruction count of the program are different
B. Because the clock rate of AMD FX is higher
C. Because the CPI of Core i7 is better
D. Because the clock rate of AMD FX is higher and CPI of Core i7 is better
E. None of the above

Amdahl's Law on Multicore Architectures

- Regarding Amdahl's Law on multicore architectures, how many of the following statements is/are correct?
  ① If we have unlimited parallelism, the performance of each parallel piece does not matter as long as the performance slowdown in each piece is bounded
  ② With unlimited amount of parallel hardware units, single-core performance does not matter anymore
  ③ With unlimited amount of parallel hardware units, the maximum speedup will be bounded by the fraction of parallel parts
  ④ With unlimited amount of parallel hardware units, the effect of scheduling and data exchange overhead is minor
  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

How programmer affects performance?

- Performance equation consists of the following three factors
  ① IC
  ② CPI
  ③ CT

  How many can a **programmer** affect?

  A. 0
  B. 1
  C. 2
  D. 3

Demo — programmer & performance

A

```
for(i = 0; i < ARRAY_SIZE; i++)
{
  for(j = 0; j < ARRAY_SIZE; j++)
  {
    c[i][j] = a[i][j]+b[i][j];
  }
}
```

B

```
for(j = 0; j < ARRAY_SIZE; j++)
{
  for(i = 0; i < ARRAY_SIZE; i++)
  {
    c[i][j] = a[i][j]+b[i][j];
  }
}
```

- How many of the following make(s) the performance of A better than B?
  - ① IC
  - ② CPI
  - ③ CT
  - A. 0
  - B. 1
  - C. 2
  - D. 3

Fair comparison

- How many of the following comparisons are fair?
  ① Comparing the frame rates of Halo 5 on AMD RyZen 1600X and civilization on Intel Core i7 7700X
  ② Using bit torrent to compare the network throughput on two machines
  ③ Comparing the frame rates of Halo 5 using medium settings on AMD RyZen 1600X and low settings on Intel Core i7 7700X
  ④ Using the peak floating point performance to judge the gaming performance of machines using AMD RyZen 1600X and Intel Core i7 7700X

A. 0
B. 1
C. 2
D. 3
E. 4

Locality

- Which description about locality of arrays `sum` and A in the following code is the most accurate?

```
for(i = 0; i< 100000; i++)
{
     sum[i%10] += A[i];
}
```

A. Access of A has temporal locality, `sum` has spatial locality
B. Both A and `sum` have temporal locality, and sum also has spatial locality
C. Access of A has spatial locality, `sum` has temporal locality
D. Both A and `sum` have spatial locality
E. Both A and `sum` have spatial locality, and sum also has temporal locality

3Cs and A, B, C

- Regarding 3Cs: compulsory, conflict and capacity misses and A, B, C: associativity, block size, capacity
  How many of the following are correct?
  ① Increasing associativity can reduce conflict misses
  ② Increasing associativity can reduce hit time
  ③ Increasing block size can increase the miss penalty
  ④ Increasing block size can reduce compulsory misses
  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

intel Core i7

- L1 data (D-L1) cache configuration of Core i7
  - Size 32KB, 8-way set associativity, 64B block
  - Assume 64-bit memory address
  - Which of the following is NOT correct?

A. Tag is 52 bits

B. Index is 6 bits

C. Offset is 6 bits

D. The cache has 128 sets

Virtual indexed, physical tagged cache limits the cache size

- If you want to build a virtual indexed, physical tagged cache with 32KB capacity, which of the following configuration is possible? Assume the system use 4K pages.
  A. 32B blocks, 2-way
  B. 32B blocks, 4-way
  C. 64B blocks, 4-way
  D. 64B blocks, 8-way

When we have virtual memory...

- In a modern x86-64 processor supports virtual memory through, how many memory accesses can an instruction incur?
    A. 2
    B. 4
    C. 6
    D. 8
    E. More than 10

Speedup of Y over X

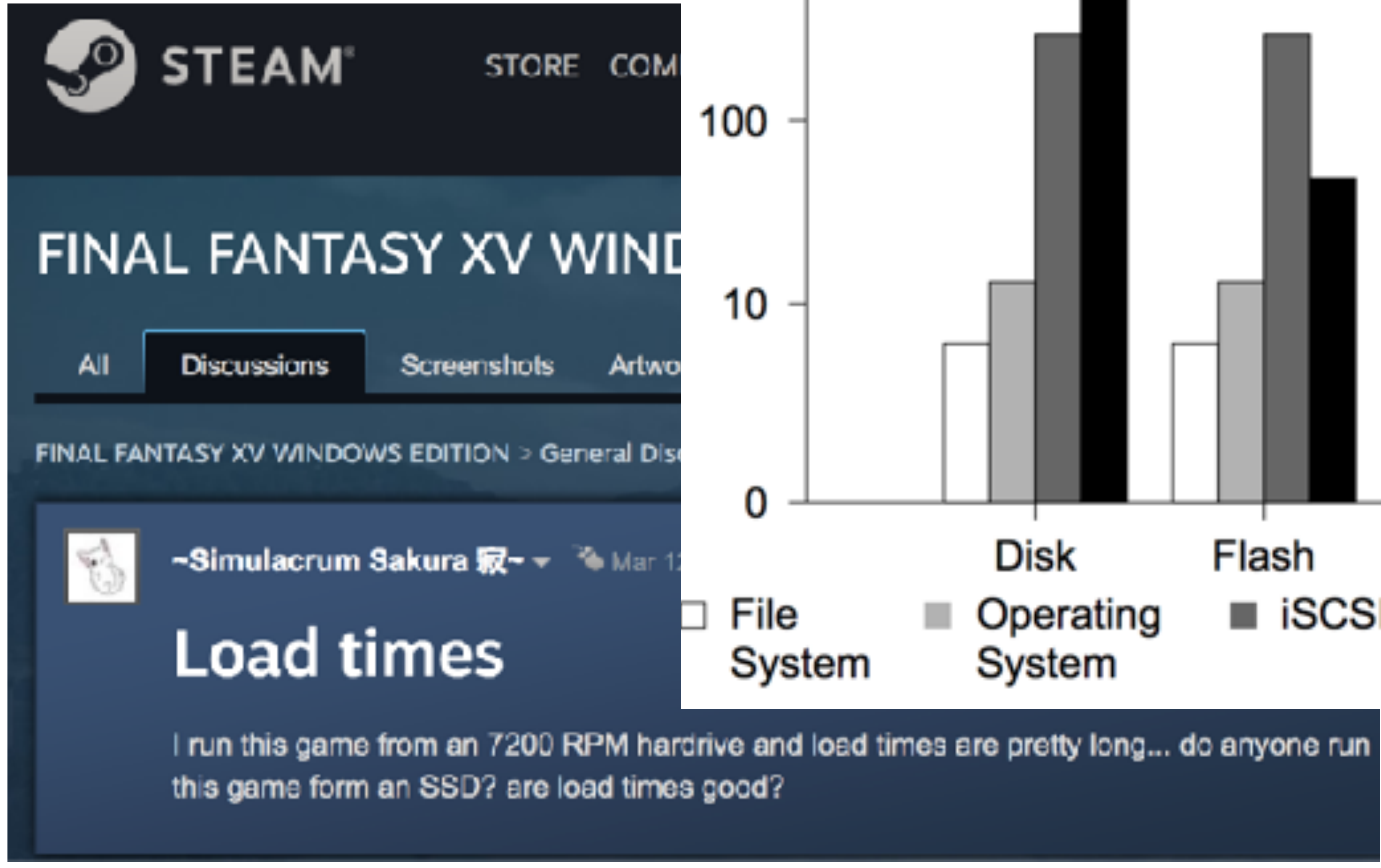
- Consider the same program on the following two machines, X and Y. By how much Y is faster than X?

	Clock Rate	Instructions	Percentage of Type-A Insts.	CPI of Type-A Insts.	Percentage of Type-B Insts.	CPI of Type-B Insts.	Percentage of Type-C Insts.	CPI of Type-C Insts.
Machine X	3 GHz	500000000	20%	8	20%	4	60%	1
Machine Y	5 GHz	500000000	20%	13	20%	4	60%	1

A. 0.2
B. 0.25
C. 0.8
D. 1.25
E. No changes

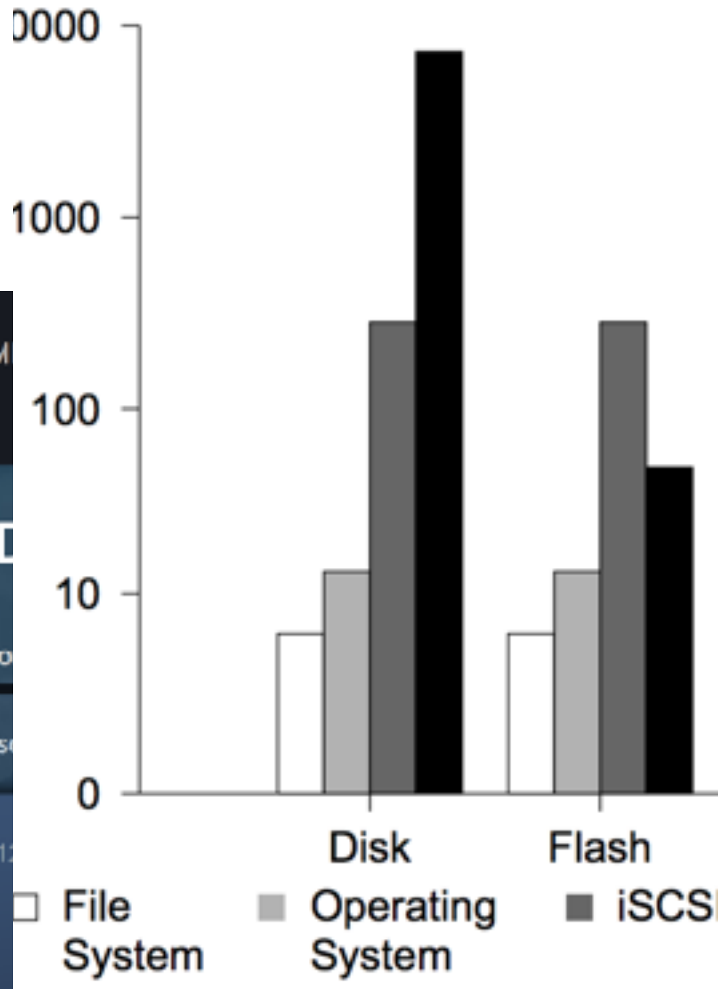
Practicing Amdahl's Law (2)

- Final Fantasy XV spends lots of time loading a map — within which period that 95% of the time on the accessing the H.D.D., the rest in the operating system, file system and the I/O protocol. If we replace the H.D.D. with a flash drive, which provides 100x faster access time and a better processor to accelerate the software overhead by 2x. By how much can we speed up the map loading process?
    A. ~7x
    B. ~10x
    C. ~17x
    D. ~29x
    E. ~100x



Load times

I run this game from an 7200 RPM hardrive and load times are pretty long... do anyone run this game form an SSD? are load times good?



Amdahl's Law on Multicore Architectures

- Regarding Amdahl's Law on multicore architectures, how many of the following statements is/are correct?
  ① If we have unlimited parallelism, the performance of each parallel piece does not matter as long as the performance slowdown in each piece is bounded
  ② With unlimited amount of parallel hardware units, single-core performance does not matter anymore
  ③ With unlimited amount of parallel hardware units, the maximum speedup will be bounded by the fraction of parallel parts
  ④ With unlimited amount of parallel hardware units, the effect of scheduling and data exchange overhead is minor
  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

AMD Phenom II

- D-L1 Cache configuration of AMD Phenom II
  - Size 64KB, 2-way set associativity, 64B block, LRU policy, write-allocate, write-back, and assuming 64-bit address.

```
int a[16384], b[16384], c[16384];
/* c = 0x10000, a = 0x20000, b = 0x30000 */
for(i = 0; i < 512; i++) {
    c[i] = a[i] + b[i];
    //load a, b, and then store to c
}
```

How many of the cache misses are **conflict** misses?

A. 6.25%

B. 66.67%

C. 68.75%

D. 93.75%

E. 100%

The result of `sizeof(struct student)`

- Consider the following data structure:

```
struct student {
    int id;
    double *homework;
    int participation;
    double midterm;
    double average;
};
```

What's the output of

`printf(“%lu\n”,sizeof(struct student))?`

A. 20

B. 28

C. 32

D. 36

E. 40

What kind(s) of misses can matrix transpose remove?

- By transposing a matrix, the performance of matrix multiplication can be further improved. What kind(s) of cache misses does matrix transpose help to remove?

Block

```
for(i = 0; i < ARRAY_SIZE; i+=(ARRAY_SIZE/n)) {
  for(j = 0; j < ARRAY_SIZE; j+=(ARRAY_SIZE/n)) {
    for(k = 0; k < ARRAY_SIZE; k+=(ARRAY_SIZE/n)) {
      for(ii = i; ii < i+(ARRAY_SIZE/n); ii++)
        for(jj = j; jj < j+(ARRAY_SIZE/n); jj++)
          for(kk = k; kk < k+(ARRAY_SIZE/n); kk++)
            c[ii][jj] += a[ii][kk]*b[kk][jj];
    }
  }
}
```

Block + Transpose

```
// Transpose matrix b into b_t
for(i = 0; i < ARRAY_SIZE; i+=(ARRAY_SIZE/n)) {
  for(j = 0; j < ARRAY_SIZE; j+=(ARRAY_SIZE/n)) {
      b_t[i][j] += b[j][i];
  }
}


for(i = 0; i < ARRAY_SIZE; i+=(ARRAY_SIZE/n)) {
  for(j = 0; j < ARRAY_SIZE; j+=(ARRAY_SIZE/n)) {
    for(k = 0; k < ARRAY_SIZE; k+=(ARRAY_SIZE/n)) {
      for(ii = i; ii < i+(ARRAY_SIZE/n); ii++)
        for(jj = j; jj < j+(ARRAY_SIZE/n); jj++)
          for(kk = k; kk < k+(ARRAY_SIZE/n); kk++
            // Compute on b_t
            c[ii][jj] += a[ii][kk]*b_t[jj][kk];
    }
  }
}
```

A. Compulsory miss

B. Capacity miss

C. Conflict miss

D. Capacity & conflict miss

E. Compulsory & conflict miss

What data structure is performing better

	Array of objects	object of arrays
	`struct grades { int id; double *homework; double average; };`	`struct grades { int *id; double **homework; double *average; };`
average of each homework	`for(i=0;i<homework_items; i++) { gradesheet[total_number_students].homework[i] = 0.0; for(j=0;j<total_number_students;j++) gradesheet[total_number_students].homework[i] +=gradesheet[j].homework[i]; gradesheet[total_number_students].homework[i] /= (double)total_number_students; }`	`for(i = 0;i < homework_items; i++) { gradesheet.homework[i][total_number_students] = 0.0; for(j = 0; j <total_number_students;j++) { gradesheet.homework[i][total_number_students] += gradesheet.homework[i][j]; } gradesheet.homework[i][total_number_students] /= total_number_students; }`

- Considering your workload would like to calculate the average score of **one of the homework** for **all students**, which data structure would deliver better performance?
  A. Array of objects
  B. Object of arrays

The impact of control hazards

- Assuming that we have an application with 20% of branch instructions and the instruction stream incurs no data hazards. When there is a branch, we disable the instruction fetch and insert no-ops until we can determine the PC. What's the average CPI if we execute this program on the 5-stage RISC-V pipeline?
  A. 1
  B. 1.2
  C. 1.4
  D. 1.6
  E. 1.8

Why can’t we proceed without stalls/no-ops?

- How many of the following statements are true regarding why we have to stall for each branch in the current pipeline processor
  ① The target address when branch is taken is not available for instruction fetch stage of the next cycle
  ② The target address when branch is not-taken is not available for instruction fetch stage of the next cycle
  ③ The branch outcome cannot be decided until the comparison result of ALU is not out
  ④ The next instruction needs the branch instruction to write back its result
  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

Which of the following schemes can help Athlon 64?

- How many of the following schemes mentioned in "improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers" would help AMD Phenom II for the code in vector addition code?
  - ① Missing cache
  - ② Victim cache
  - ③ Prefetch
  - ④ Stream buffer
  - A. 0
  - B. 1
  - C. 2
  - D. 3
  - E. 4

Sample short answer questions (< 30 words)

- What are the limitations of compiler optimizations? Can you list two?
- Please define Amdahl's Law and explain each term in it
- Please define the CPU performance equation and explain each term.
- Can you list two things affecting each term in the performance equation?
- What's the difference between latency and throughput? When should you use latency or throughput to judge performance?
- What's "benchmark" suite? Why is it important?
- Why TFLOPS or inferences per second is not a good metrics?

Amdahl’s Law for multiple optimizations

- Assume that memory access takes 30% of execution time.
  - Cache can speedup 80% of memory operation by a factor of 4
  - L2 cache can speedup 50% of the remaining 20% by a factor of 2
- What’s the total speedup?

Performance evaluation with cache

- Consider the following cache configuration on RISC-V processor:

	I-L1	D-L1	L2	DRAM
size	32K	32K	256K	Big enough
block size	64 Bytes	64 Bytes	64 Bytes	4KB pages
associativity	2-way	2-way	8-way	
access time	1 cycle (no penalty if it's a hit)	1 cycle (no penalty if it's a hit)	10 cycles	100 cycles
local miss rate	2%	10%, 20% dirty	15% (i.e., 15% of L1 misses, also miss in the L2), 30% dirty	
Write policy	N/A	Write-back, write allocate		
Replacement	LRU replacement policy			

The application has 20% branches, 10% loads/stores, 70% integer instructions.
Assume that TLB miss rate is 2% and it requires 100 cycles to handle a TLB miss. Also assume that the branch predictor has a hit rate of 87.5%, what's the CPI of branch, L/S, and integer instructions? What is the average CPI?

Cache simulation

- The processor has a 8KB, 256B blocked, 2-way L1 cache. Consider the following code:

  ```
  for(i=0;i<256;i++) {
      a[i] = b[i] + c[i];
  // load a[i] and load b[i], store to c[i]
  // &a[0] = 0x10000, &b[0] = 0x20000, &c[0] = 0x30000
  }
  ```

- What's the total miss rate? How many of the misses are compulsory misses? How many of the misses are conflict misses?
- How can you improve the cache performance of the above code through changing hardware?
- How can you improve the performance **without** changing hardware?

Announcement

- Midterm
  - Release tonight after the assignment deadline, turn in before **Friday 11:59:00pm**
  - You can only open it **once** and you have to finish **a total of 30 questions within 80 minutes**.
  - You may open book, but you have to bare the risks of not being able to finish them
- Project is up — check the website
- Assignment #3 due tonight
- Attendance
  - The attendance throughout the quarter count as one assignment
  - You only need to answer 50% of the Zoom polls to receive full credits
    - Please don't email me for absence — we count only 50% to give you flexibility
    - If you just login but never answer questions, you won't receive any.
- Office Hours on Zoom (the office hour link, not the lecture one)
  - Hung-Wei/Prof. Usagi: M 8p-10p (make up for the last week), W 2p-3p
  - Quan Fan: F 1p-3p

Computer
Science &
Engineering

203

つづく