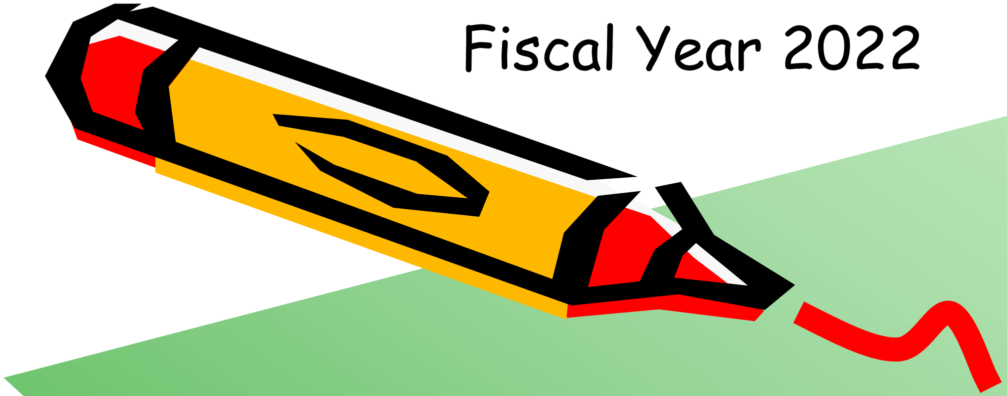
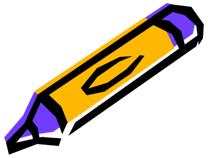


Fiscal Year 2022

Course number: CSC.T433
School of Computing,
Graduate major in Computer Science

Advanced Computer Architecture

Single-cycle processor, and Memory Hierarchy Design



www.arch.cs.titech.ac.jp/lecture/ACA/
Room No.W831, HyFlex
Mon 13:45-15:25, Thr 13:45-15:25

Kenji Kise, Department of Computer Science
kise _at_ c.titech.ac.jp

The past, present, and future of the world's most important device

The Best of Hands On › The aspirational delight of our top DIY projects P.18

Stronger Software, Fewer Bugs › The case for functional programming P.40

The Next Step in Worker Productivity › EEG Brain Scans P.46

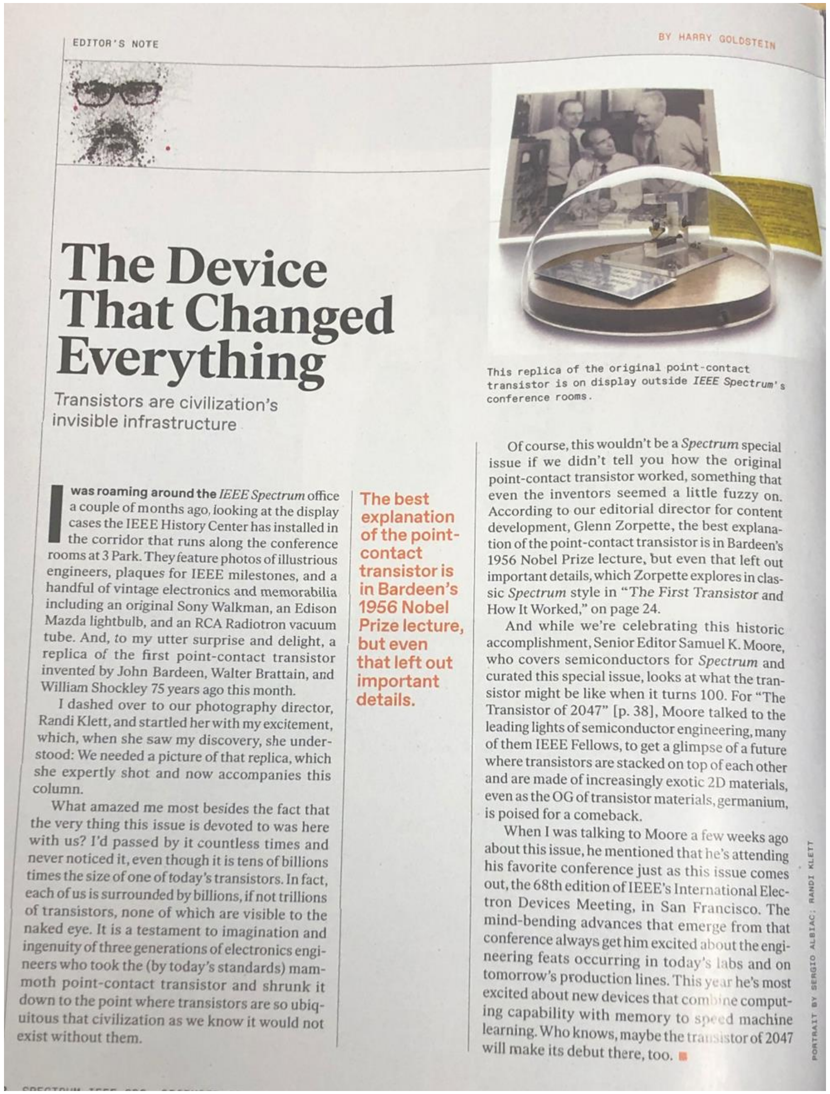
FOR THE TECHNOLOGY INSIDER

DECEMBER 2022

IEEE Spectrum

The Transistor at 75

The Past, Present, and Future of the World's Most Important Device



EDITOR'S NOTE

BY HARRY GOLDSTEIN

The Device That Changed Everything

Transistors are civilization's invisible infrastructure

This replica of the original point-contact transistor is on display outside *IEEE Spectrum*'s conference rooms.

I was roaming around the *IEEE Spectrum* office a couple of months ago, looking at the display cases the IEEE History Center has installed in the corridor that runs along the conference rooms at 3 Park. They feature photos of illustrious engineers, plaques for IEEE milestones, and a handful of vintage electronics and memorabilia including an original Sony Walkman, an Edison Mazda lightbulb, and an RCA Radiotron vacuum tube. And, to my utter surprise and delight, a replica of the first point-contact transistor invented by John Bardeen, Walter Brattain, and William Shockley 75 years ago this month.

I dashed over to our photography director, Randi Klett, and startled her with my excitement, which, when she saw my discovery, she understood: We needed a picture of that replica, which she expertly shot and now accompanies this column.

What amazed me most besides the fact that the very thing this issue is devoted to was here with us? I'd passed by it countless times and never noticed it, even though it is tens of billions times the size of one of today's transistors. In fact, each of us is surrounded by billions, if not trillions of transistors, none of which are visible to the naked eye. It is a testament to imagination and ingenuity of three generations of electronics engineers who took the (by today's standards) mammoth point-contact transistor and shrunk it down to the point where transistors are so ubiquitous that civilization as we know it would not exist without them.

The best explanation of the point-contact transistor is in Bardeen's 1956 Nobel Prize lecture, but even that left out important details.

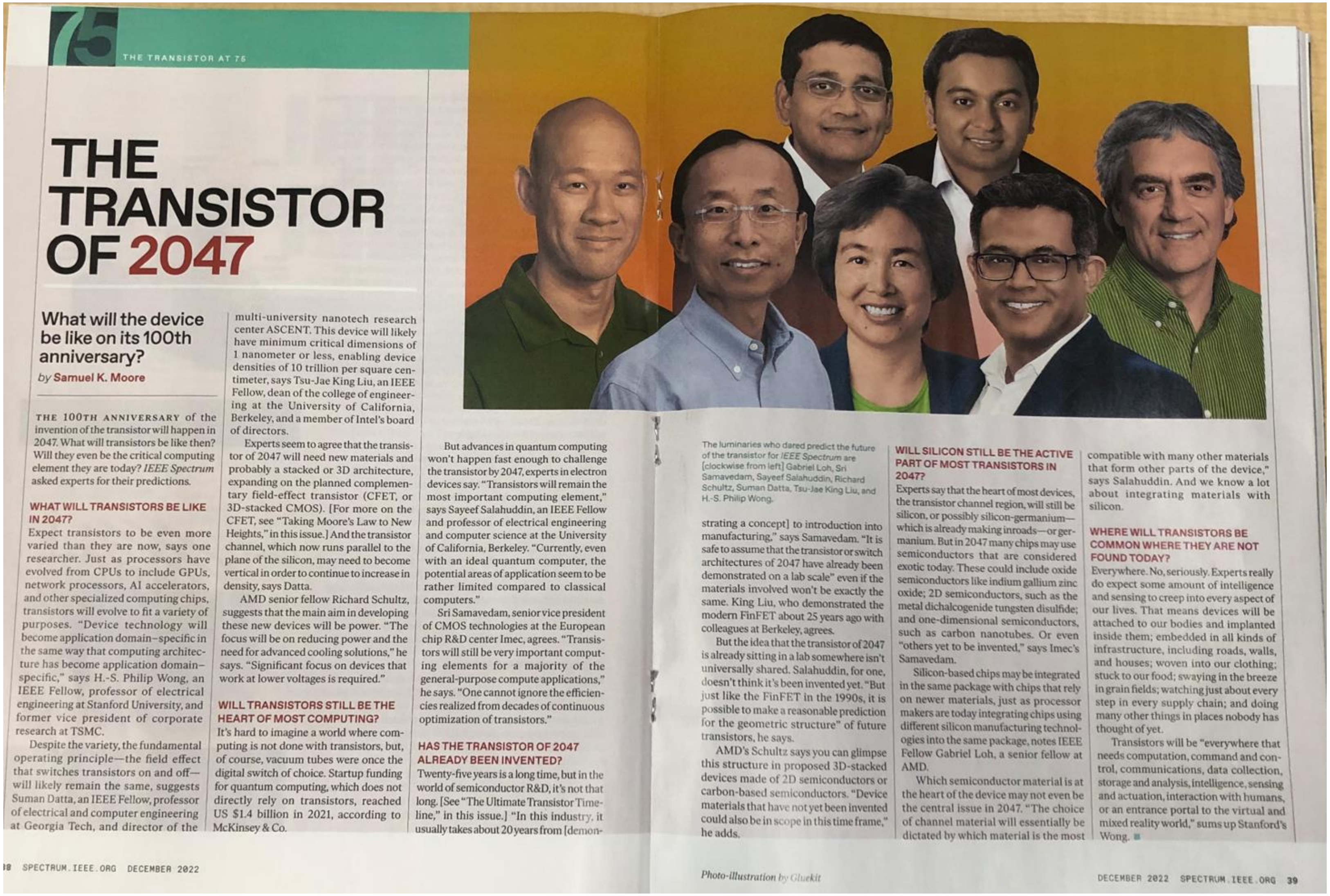
Of course, this wouldn't be a *Spectrum* special issue if we didn't tell you how the original point-contact transistor worked, something that even the inventors seemed a little fuzzy on. According to our editorial director for content development, Glenn Zorpette, the best explanation of the point-contact transistor is in Bardeen's 1956 Nobel Prize lecture, but even that left out important details, which Zorpette explores in classic *Spectrum* style in "*The First Transistor* and How It Worked," on page 24.

And while we're celebrating this historic accomplishment, Senior Editor Samuel K. Moore, who covers semiconductors for *Spectrum* and curated this special issue, looks at what the transistor might be like when it turns 100. For "The Transistor of 2047" [p. 38], Moore talked to the leading lights of semiconductor engineering, many of them IEEE Fellows, to get a glimpse of a future where transistors are stacked on top of each other and are made of increasingly exotic 2D materials, even as the OG of transistor materials, germanium, is poised for a comeback.

When I was talking to Moore a few weeks ago about this issue, he mentioned that he's attending his favorite conference just as this issue comes out, the 68th edition of IEEE's International Electron Devices Meeting, in San Francisco. The mind-bending advances that emerge from that conference always get him excited about the engineering feats occurring in today's labs and on tomorrow's production lines. This year he's most excited about new devices that combine computing capability with memory to speed machine learning. Who knows, maybe the transistor of 2047 will make its debut there, too. ■

PORTRAIT BY SERGIO ALBIAC; RANDI KLETT

The past, present, and future of the world's most important device



THE TRANSISTOR OF 2047

What will the device be like on its 100th anniversary?

by Samuel K. Moore

THE 100TH ANNIVERSARY of the invention of the transistor will happen in 2047. What will transistors be like then? Will they even be the critical computing element they are today? *IEEE Spectrum* asked experts for their predictions.

WHAT WILL TRANSISTORS BE LIKE IN 2047?

Expect transistors to be even more varied than they are now, says one researcher. Just as processors have evolved from CPUs to include GPUs, network processors, AI accelerators, and other specialized computing chips, transistors will evolve to fit a variety of purposes. "Device technology will become application domain–specific in the same way that computing architecture has become application domain–specific," says H.-S. Philip Wong, an IEEE Fellow, professor of electrical engineering at Stanford University, and former vice president of corporate research at TSMC.

Despite the variety, the fundamental operating principle—the field effect that switches transistors on and off—will likely remain the same, suggests Suman Datta, an IEEE Fellow, professor of electrical and computer engineering at Georgia Tech, and director of the multi-university nanotech research center ASCENT. This device will likely have minimum critical dimensions of 1 nanometer or less, enabling device densities of 10 trillion per square centimeter, says Tsu-Jae King Liu, an IEEE Fellow, dean of the college of engineering at the University of California, Berkeley, and a member of Intel's board of directors.

Experts seem to agree that the transistor of 2047 will need new materials and probably a stacked or 3D architecture, expanding on the planned complementary field-effect transistor (CFET, or 3D-stacked CMOS). [For more on the CFET, see "Taking Moore's Law to New Heights," in this issue.] And the transistor channel, which now runs parallel to the plane of the silicon, may need to become vertical in order to continue to increase in density, says Datta.

AMD senior fellow Richard Schultz, suggests that the main aim in developing these new devices will be power. "The focus will be on reducing power and the need for advanced cooling solutions," he says. "Significant focus on devices that work at lower voltages is required."

WILL TRANSISTORS STILL BE THE HEART OF MOST COMPUTING?

It's hard to imagine a world where computing is not done with transistors, but, of course, vacuum tubes were once the digital switch of choice. Startup funding for quantum computing, which does not directly rely on transistors, reached US $1.4 billion in 2021, according to McKinsey & Co.

But advances in quantum computing won't happen fast enough to challenge the transistor by 2047, experts in electron devices say. "Transistors will remain the most important computing element," says Sayeef Salahuddin, an IEEE Fellow and professor of electrical engineering and computer science at the University of California, Berkeley. "Currently, even with an ideal quantum computer, the potential areas of application seem to be rather limited compared to classical computers."

Sri Samavedam, senior vice president of CMOS technologies at the European chip R&D center Imec, agrees. "Transistors will still be very important computing elements for a majority of the general-purpose compute applications," he says. "One cannot ignore the efficiencies realized from decades of continuous optimization of transistors."

HAS THE TRANSISTOR OF 2047 ALREADY BEEN INVENTED?

Twenty-five years is a long time, but in the world of semiconductor R&D, it's not that long. [See "The Ultimate Transistor Timeline," in this issue.] "In this industry, it usually takes about 20 years from [demonstrating a concept] to introduction into manufacturing," says Samavedam. "It is safe to assume that the transistor or switch architectures of 2047 have already been demonstrated on a lab scale" even if the materials involved won't be exactly the same. King Liu, who demonstrated the modern FinFET about 25 years ago with colleagues at Berkeley, agrees.

But the idea that the transistor of 2047 is already sitting in a lab somewhere isn't universally shared. Salahuddin, for one, doesn't think it's been invented yet. "But just like the FinFET in the 1990s, it is possible to make a reasonable prediction for the geometric structure" of future transistors, he says.

AMD's Schultz says you can glimpse this structure in proposed 3D-stacked devices made of 2D semiconductors or carbon-based semiconductors. "Device materials that have not yet been invented could also be in scope in this time frame," he adds.

WILL SILICON STILL BE THE ACTIVE PART OF MOST TRANSISTORS IN 2047?

Experts say that the heart of most devices, the transistor channel region, will still be silicon, or possibly silicon-germanium—which is already making inroads—or germanium. But in 2047 many chips may use semiconductors that are considered exotic today. These could include oxide semiconductors like indium gallium zinc oxide; 2D semiconductors, such as the metal dichalcogenide tungsten disulfide; and one-dimensional semiconductors, such as carbon nanotubes. Or even "others yet to be invented," says Imec's Samavedam.

Silicon-based chips may be integrated in the same package with chips that rely on newer materials, just as processor makers are today integrating chips using different silicon manufacturing technologies into the same package, notes IEEE Fellow Gabriel Loh, a senior fellow at AMD.

Which semiconductor material is at the heart of the device may not even be the central issue in 2047. "The choice of channel material will essentially be dictated by which material is the most compatible with many other materials that form other parts of the device," says Salahuddin. And we know a lot about integrating materials with silicon.

WHERE WILL TRANSISTORS BE COMMON WHERE THEY ARE NOT FOUND TODAY?

Everywhere. No, seriously. Experts really do expect some amount of intelligence and sensing to creep into every aspect of our lives. That means devices will be attached to our bodies and implanted inside them; embedded in all kinds of infrastructure, including roads, walls, and houses; woven into our clothing; stuck to our food; swaying in the breeze in grain fields; watching just about every step in every supply chain; and doing many other things in places nobody has thought of yet.

Transistors will be "everywhere that needs computation, command and control, communications, data collection, storage and analysis, intelligence, sensing and actuation, interaction with humans, or an entrance portal to the virtual and mixed reality world," sums up Stanford's Wong. ■

The luminaries who dared predict the future of the transistor for *IEEE Spectrum* are [clockwise from left] Gabriel Loh, Sri Samavedam, Sayeef Salahuddin, Richard Schultz, Suman Datta, Tsu-Jae King Liu, and H.-S. Philip Wong.

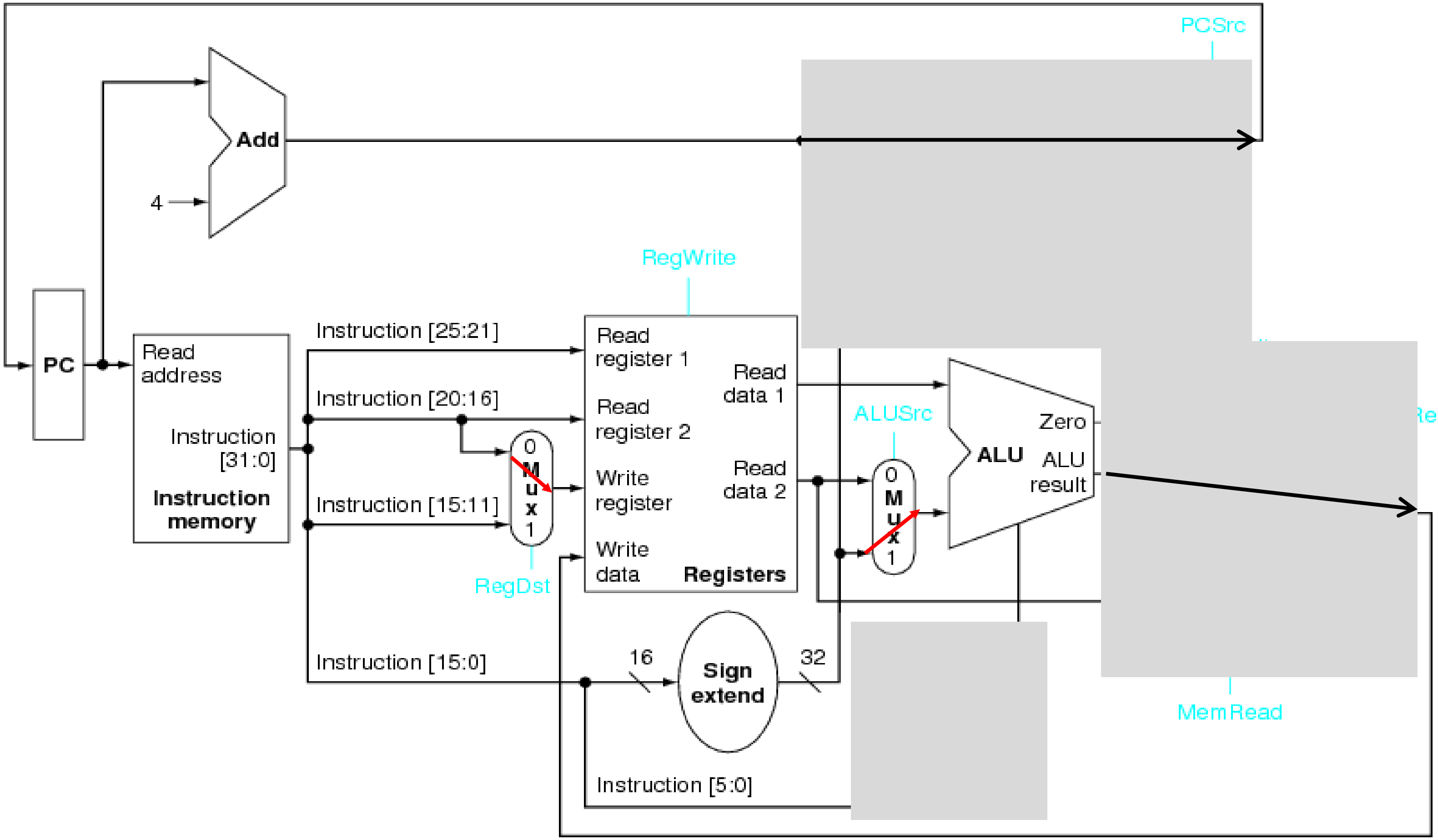
Photo-illustration by Gluekit

Datapath of processor supporting ADD and ADDI

	IR[25:21]	IR[20:16]	
op	rs	rt	16 bit immediate

I format

```
0x804  addi $9, $8, 3
```

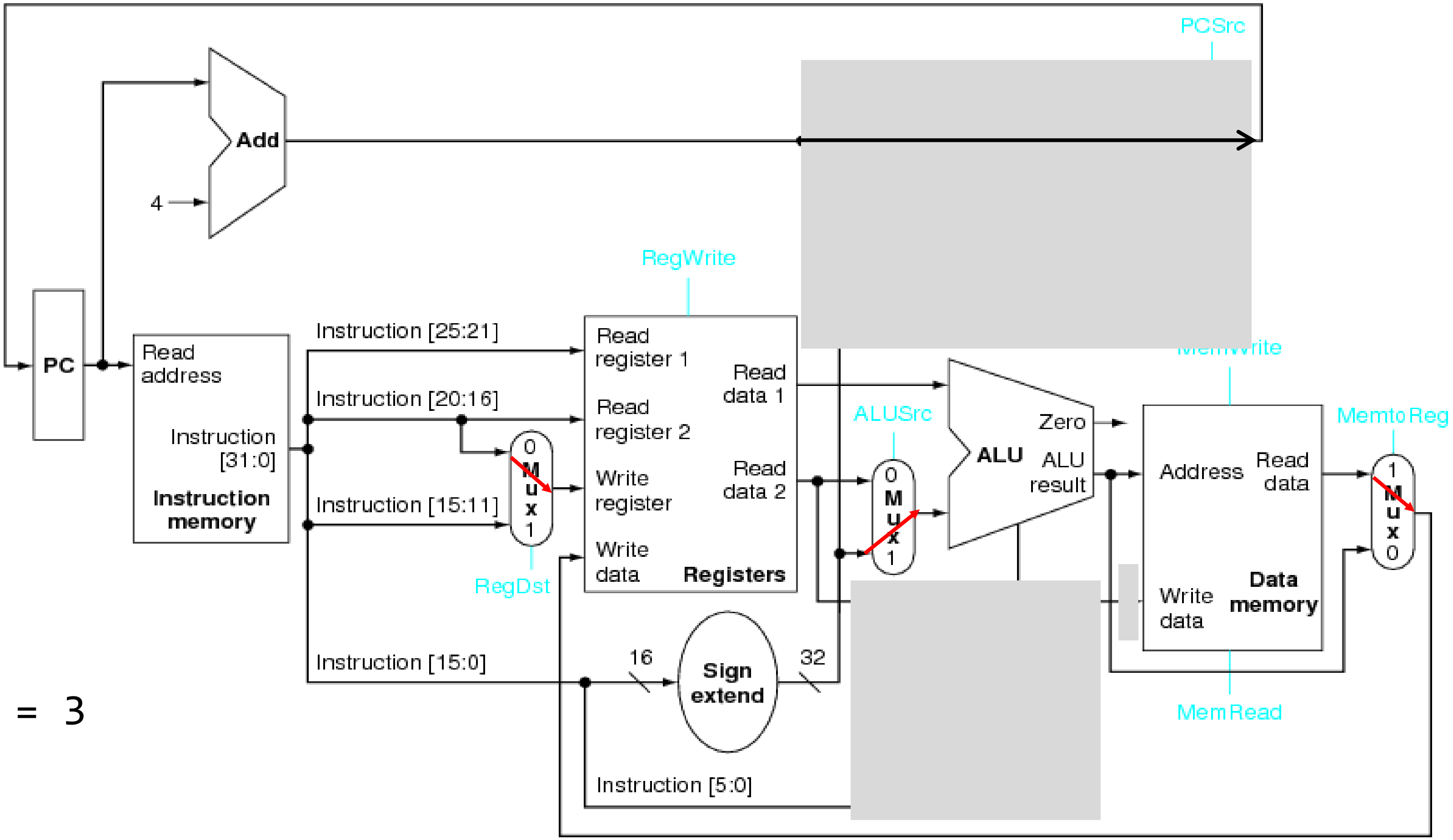


$8 = 7

Datapath of processor supporting ADD, ADDI, LW

	IR[25:21]	IR[20:16]		
op	rs	rt	16 bit immediate	I format

```
0x808  lw $10, 4($8)
```



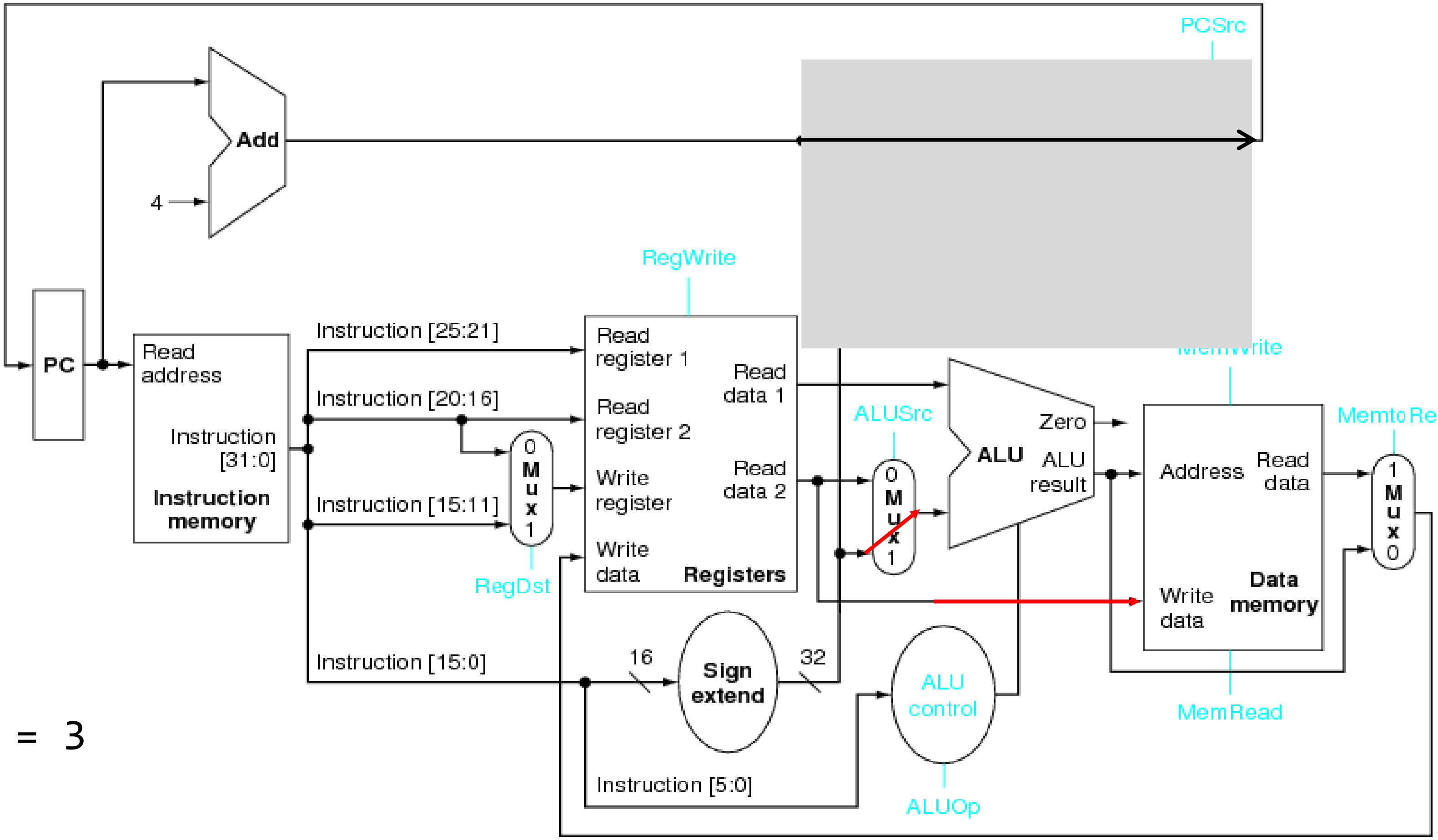
```
$8 = 12
mem[16] = 3
```

Datapath of processor supporting ADD, ADDI, LW, SW

	IR[25:21]	IR[20:16]	
op	rs	rt	16 bit immediate

I format

```
0x808  sw $10, 4($8)
```



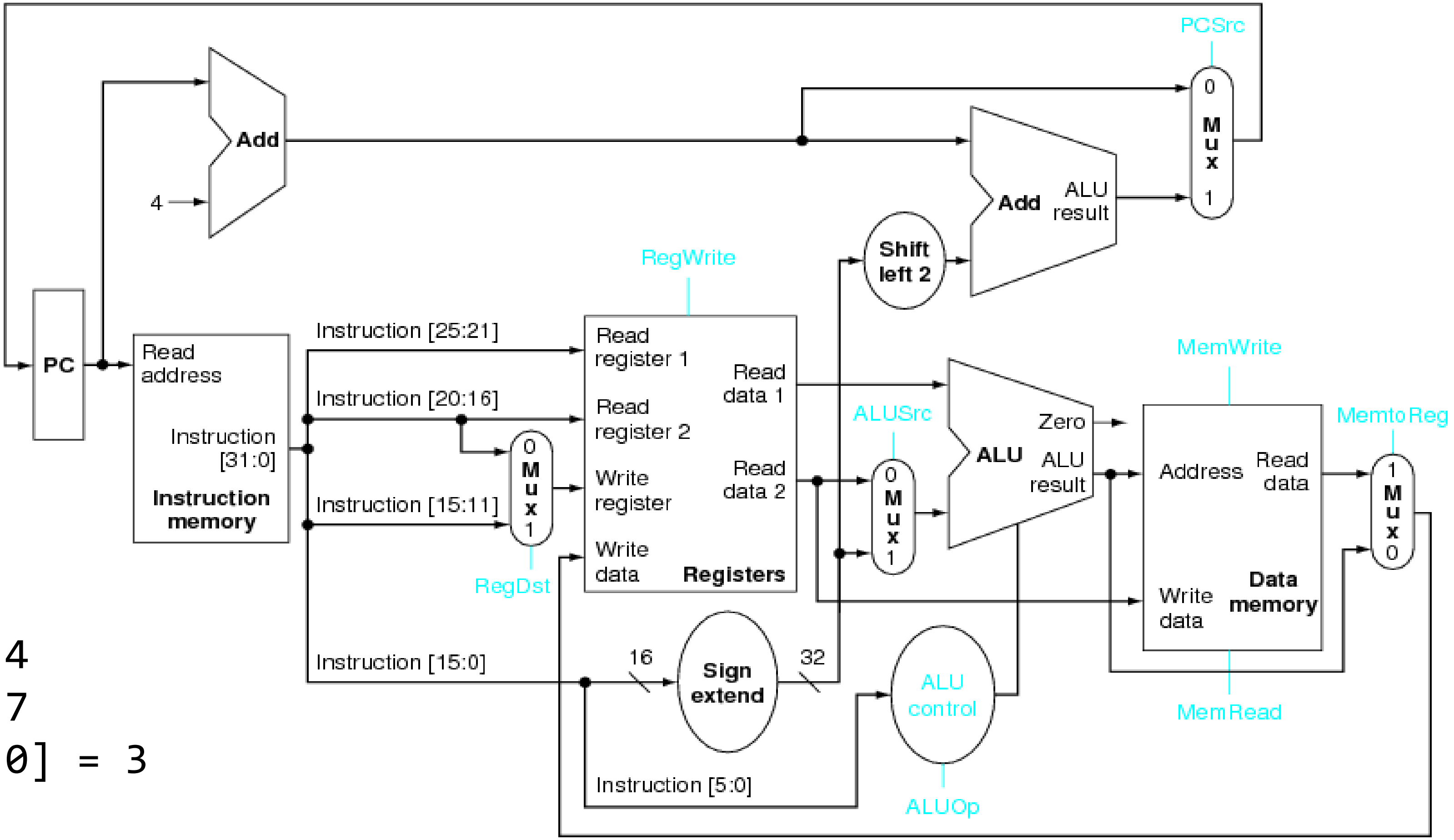
```
$8 = 12
$10 = 5
mem[16] = 3
```

Datapath of proc. supporting ADD, ADDI, LW, SW, BNE

	IR[25:21]	IR[20:16]		
op	rs	rt	16 bit immediate	I format

```
0x808  bne $10, $11, Label
```

```
0x808 bne
0x80c insn2
0x810 insn3
0x814 insn4
0x818 insn5
```

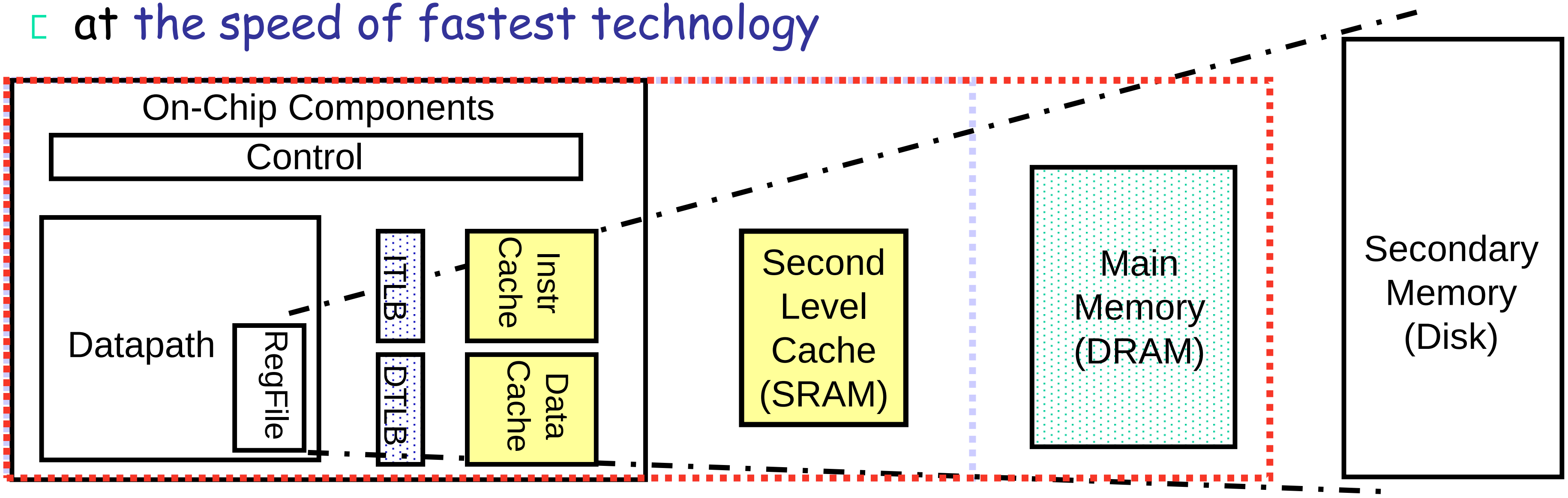


```
$10 = 4
$11 = 7
IR[15:0] = 3
```

this slide is to be used as a whiteboard

A Typical Memory Hierarchy

- By taking advantage of the principle of locality in time and space
  - Present much memory in the cheapest technology
  - at the speed of fastest technology

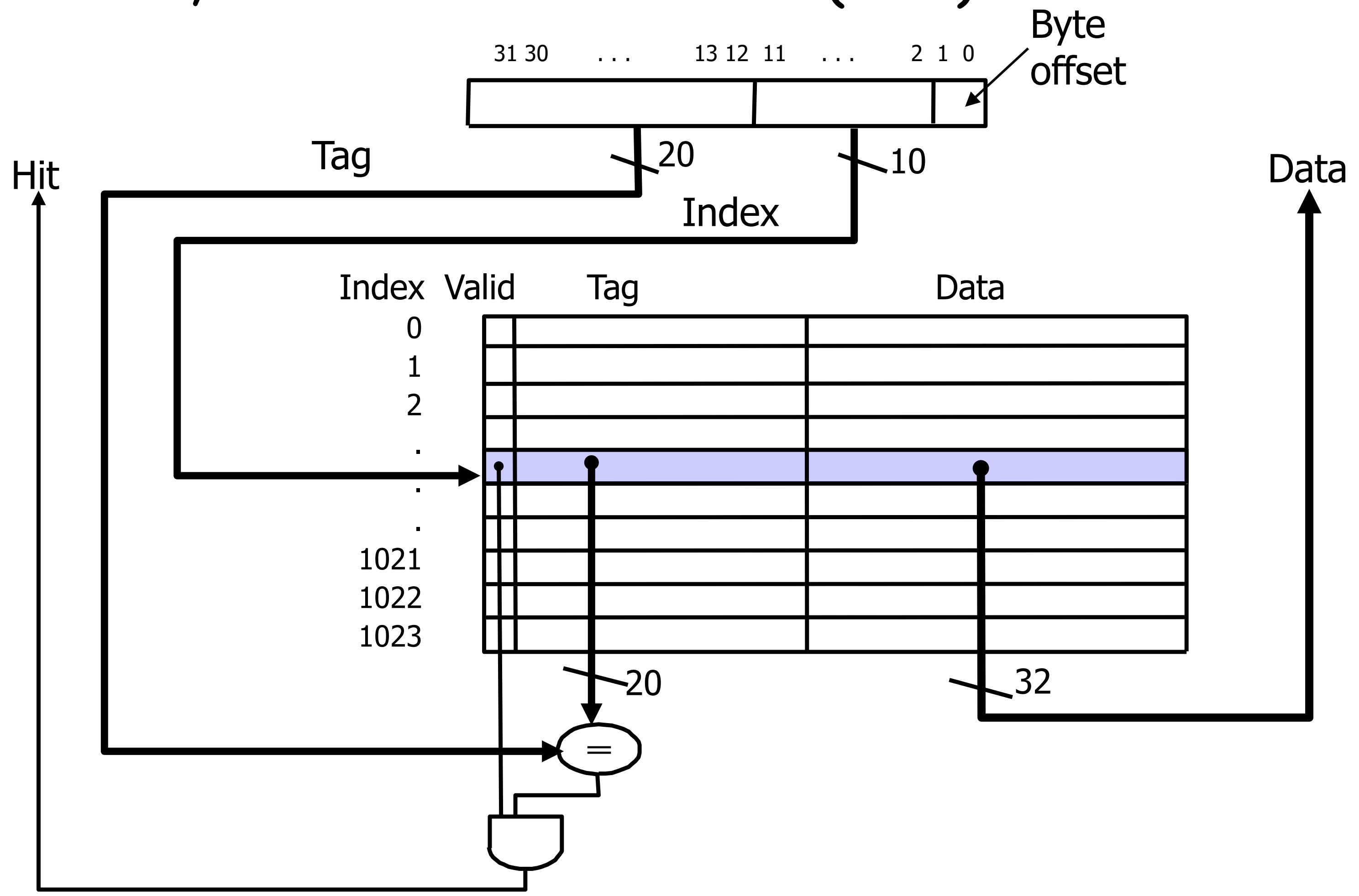


Speed (%cycles):	½'s	1's	10's	100's	1,000's
Size (bytes):	100's	K's	10K's	M's	G's to T's
Cost:	highest				lowest

TLB: Translation Lookaside Buffer

MIPS Direct Mapped Cache Example

- One word/block, cache size = 1K words (4KB)



What kind of locality are we taking advantage of?

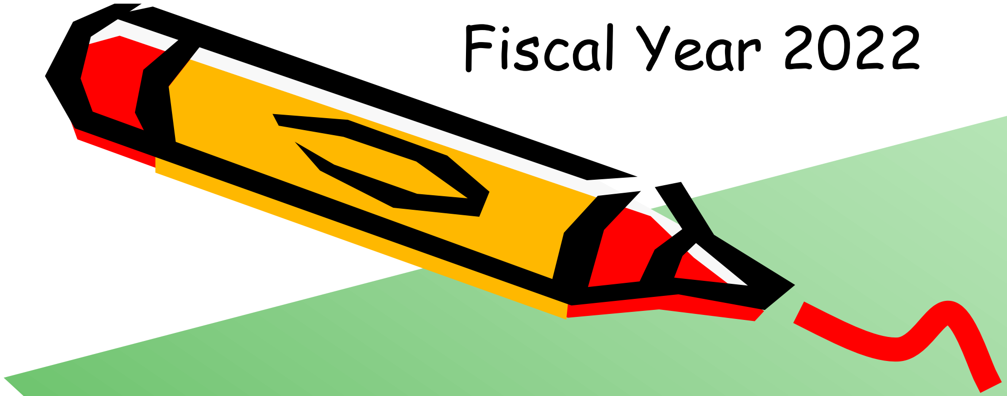
this slide is to be used as a whiteboard

Assignment 3

1. Design a single-cycle processor supporting MIPS add, addi, lw and sw instructions in Verilog HDL.
2. Verify the behavior of designed processor using following assembly code

```
add  $0,  $0,  $0   # NOP {6'h0, 5'd0, 5'd0, 5'd0, 5'd0, 6'h20}
addi $8, $0, 8      # {6'h8, 5'd0, 5'd8, 16’d8}, $8 = 8
sw   $8, 4($8)      # {6'h2b,5'd8, 5'd8, 16’d4}, mem[12] = 8
lw   $9, 4($8)      # {6'h23,5'd8, 5'd9, 16’d4}, $9 = mem[12]
addi $10, $9, 6     # {6'h8, 5'd9, 5'd10,16’h6}, $10 = $9 + 6
```

3. Submit your report in a PDF file via E-mail (kise [at] c.titech.ac.jp) by 13:00 on January 5th.
   - The report should include a block diagram, a source code in Verilog HDL, and obtained waveforms of your design.
   - E-mail title: Assignment of Advanced Computer Architecture

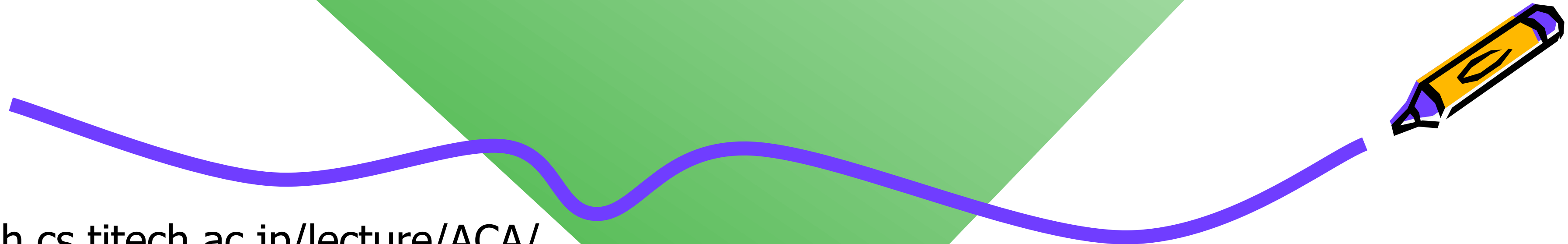


Fiscal Year 2022

Course number: CSC.T433
School of Computing,
Graduate major in Computer Science

Advanced Computer Architecture

4. Pipelining

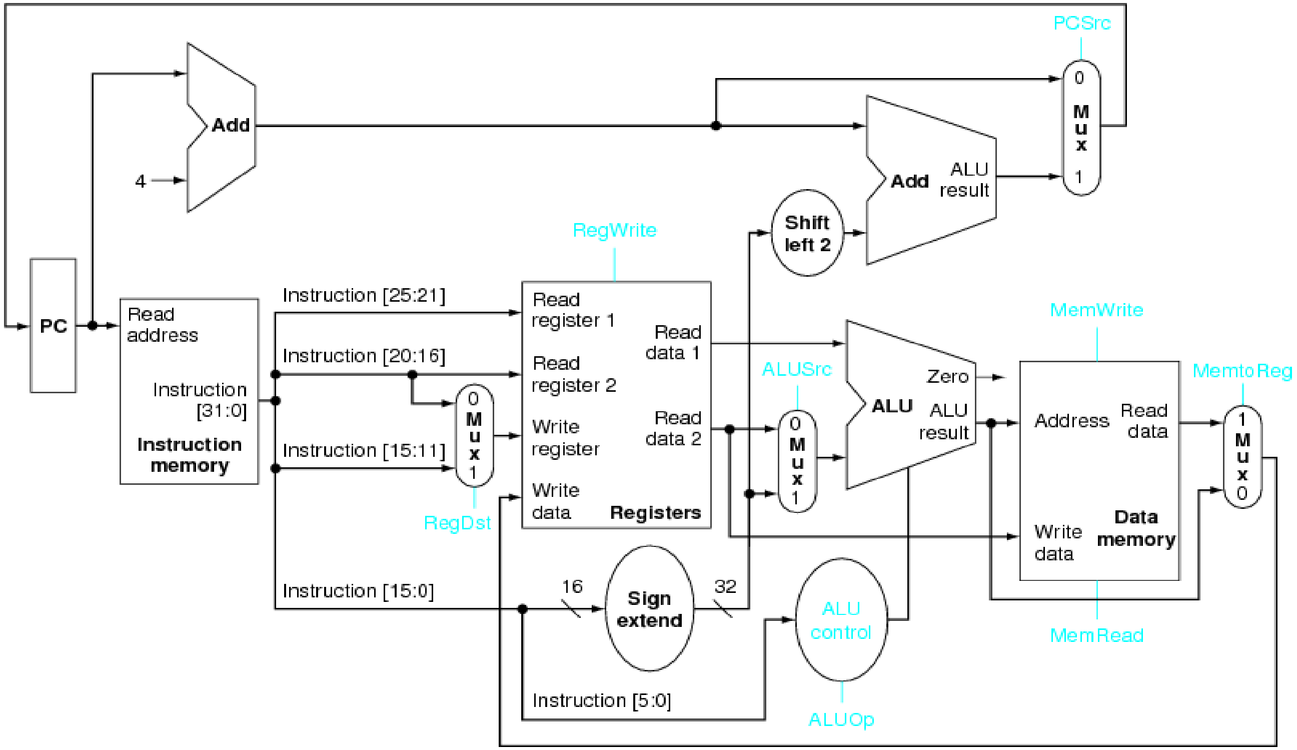


www.arch.cs.titech.ac.jp/lecture/ACA/
Room No.W831, HyFlex
Mon 13:45-15:25, Thr 13:45-15:25

Kenji Kise, Department of Computer Science
kise _at_ c.titech.ac.jp

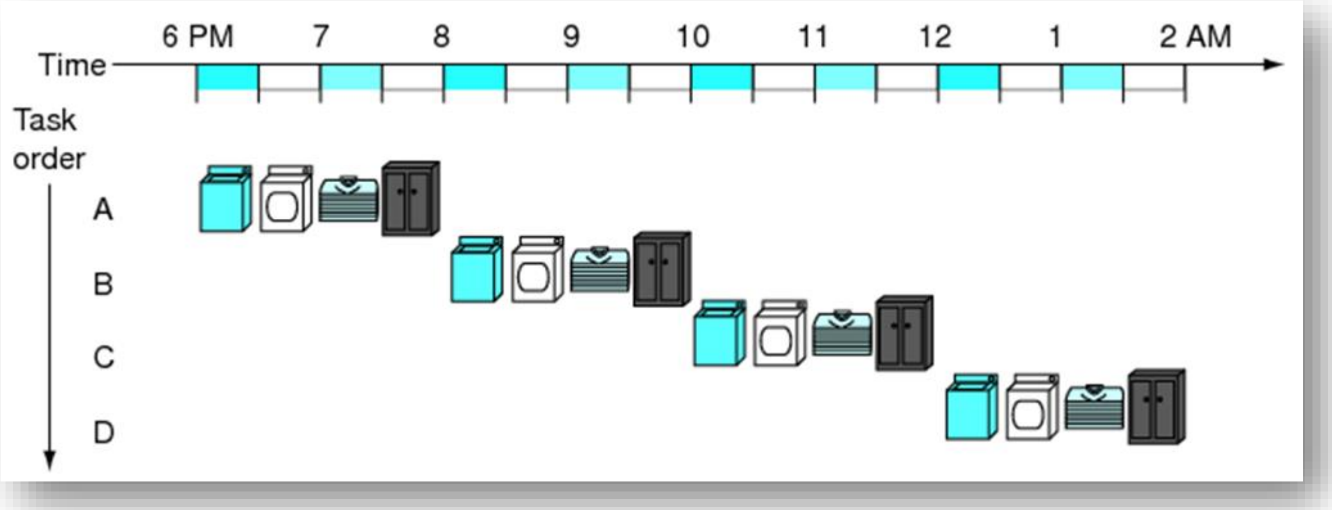
Single-cycle implementation of processors

- Single-cycle implementation also called single clock cycle implementation is the implementation in which an instruction is executed in one clock cycle. While easy to understand, it is too slow to be practical.



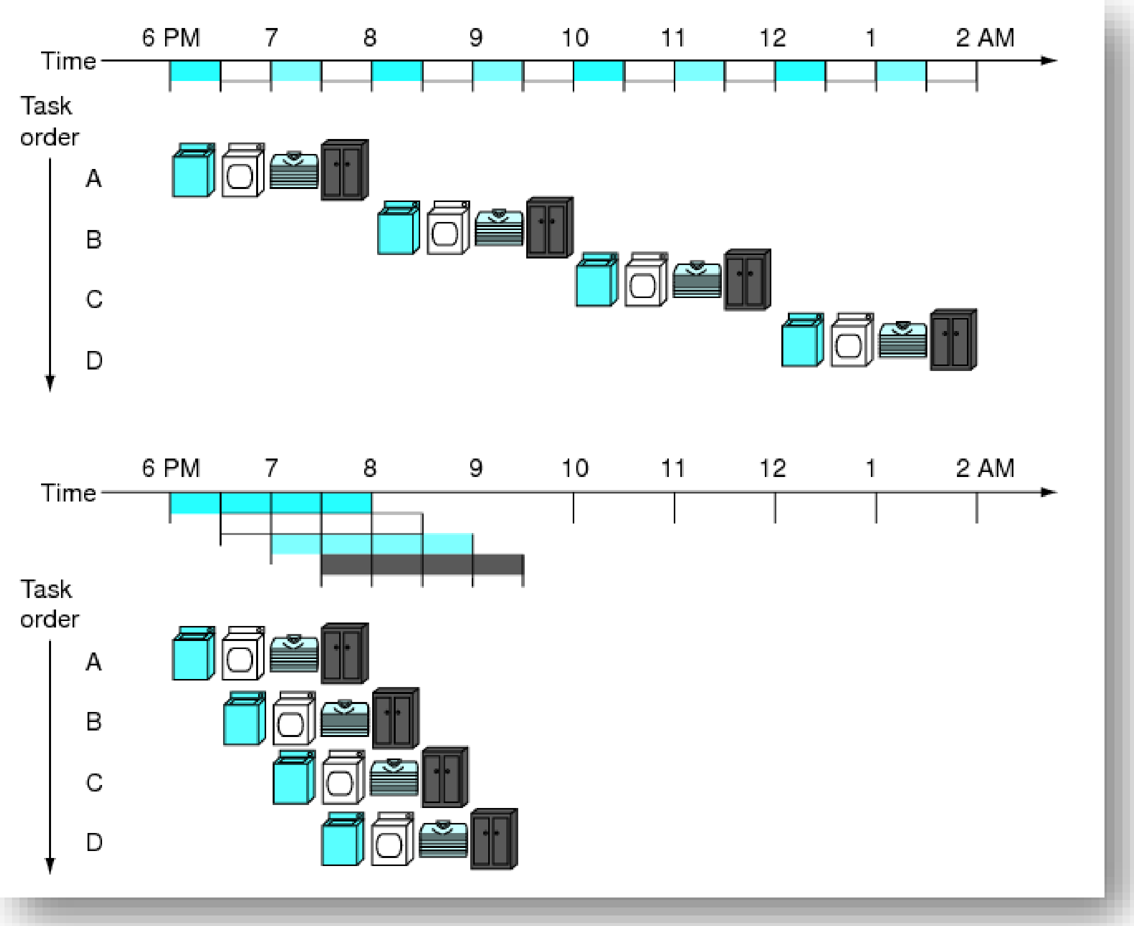
Single-cycle implementation of laundry

- (A) Ann, (B) Brian, (C) Cathy, and (D) Don each have dirty clothes to be *washed*, *dried*, *folded*, and *put away* where each takes 30 minutes.
- Cycle time is 2 hours.
- Sequential laundry takes 8 hours for 4 loads.

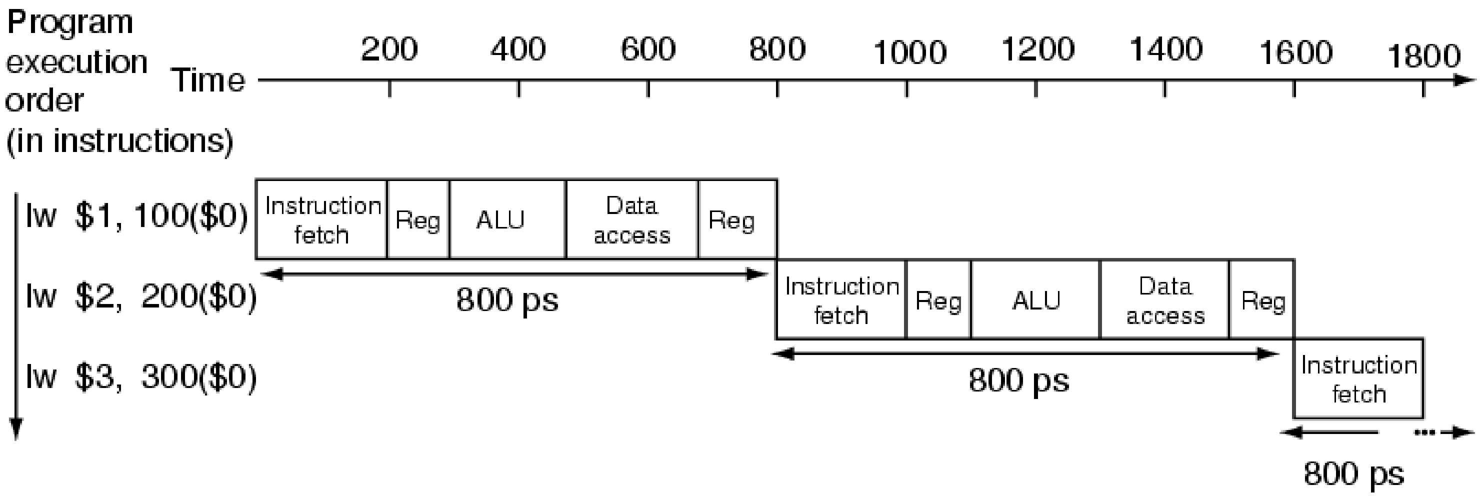


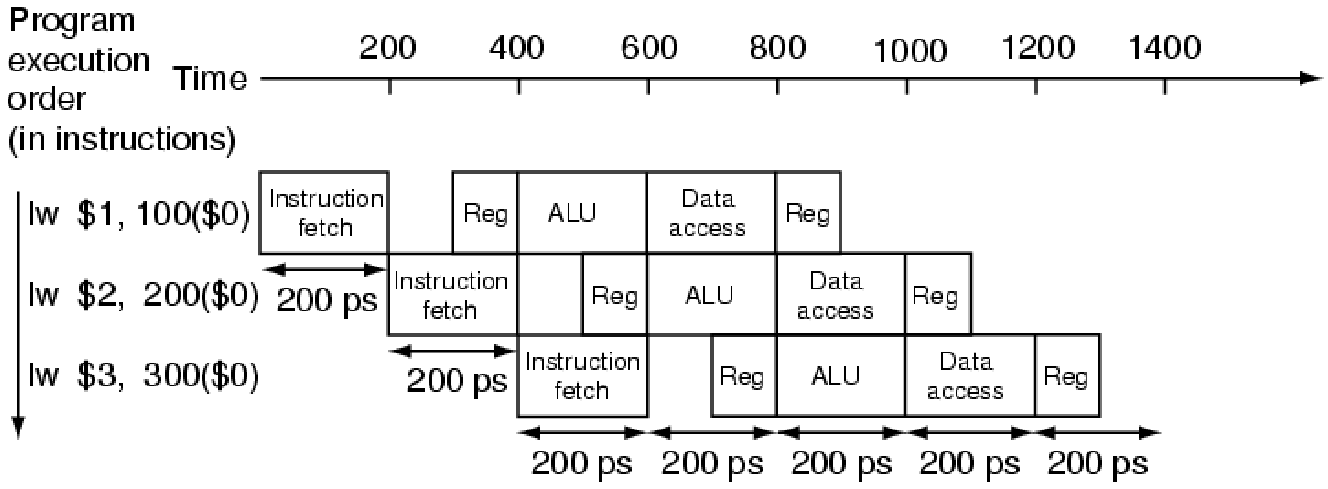
Single-cycle implementation and pipelining

- Pipelined laundry takes 3.5 hours just using the same hardware resources. Cycle time is 30 minutes.
- What is the latency of each load?



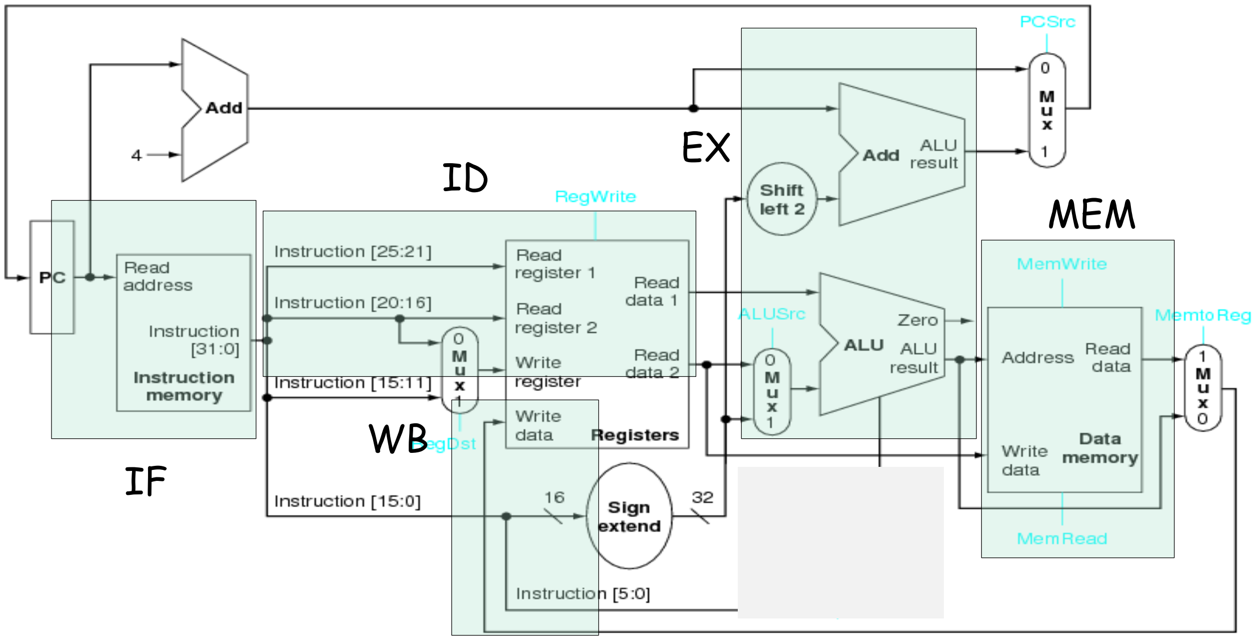
Single-cycle and pipelined processors





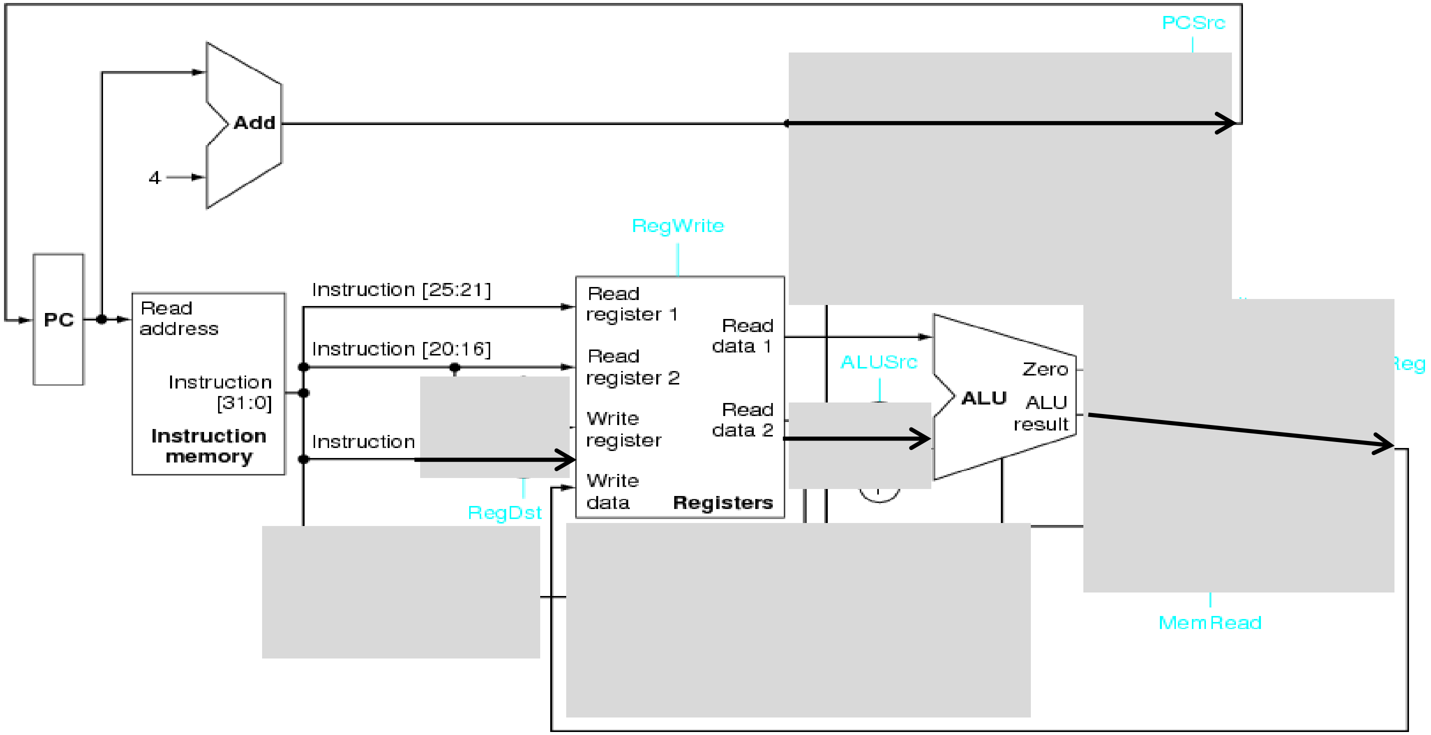
Conventional five steps (stages) of MIPS

- IF: Instruction Fetch from instruction memory
- ID: Instruction Decode and operand fetch from regfile (register file)
- EX: EXecute operation or calculate address for load/store or calculate branch condition and target address
- MEM (MA): MEMory access for load/store
- WB: Write result Back to regfile



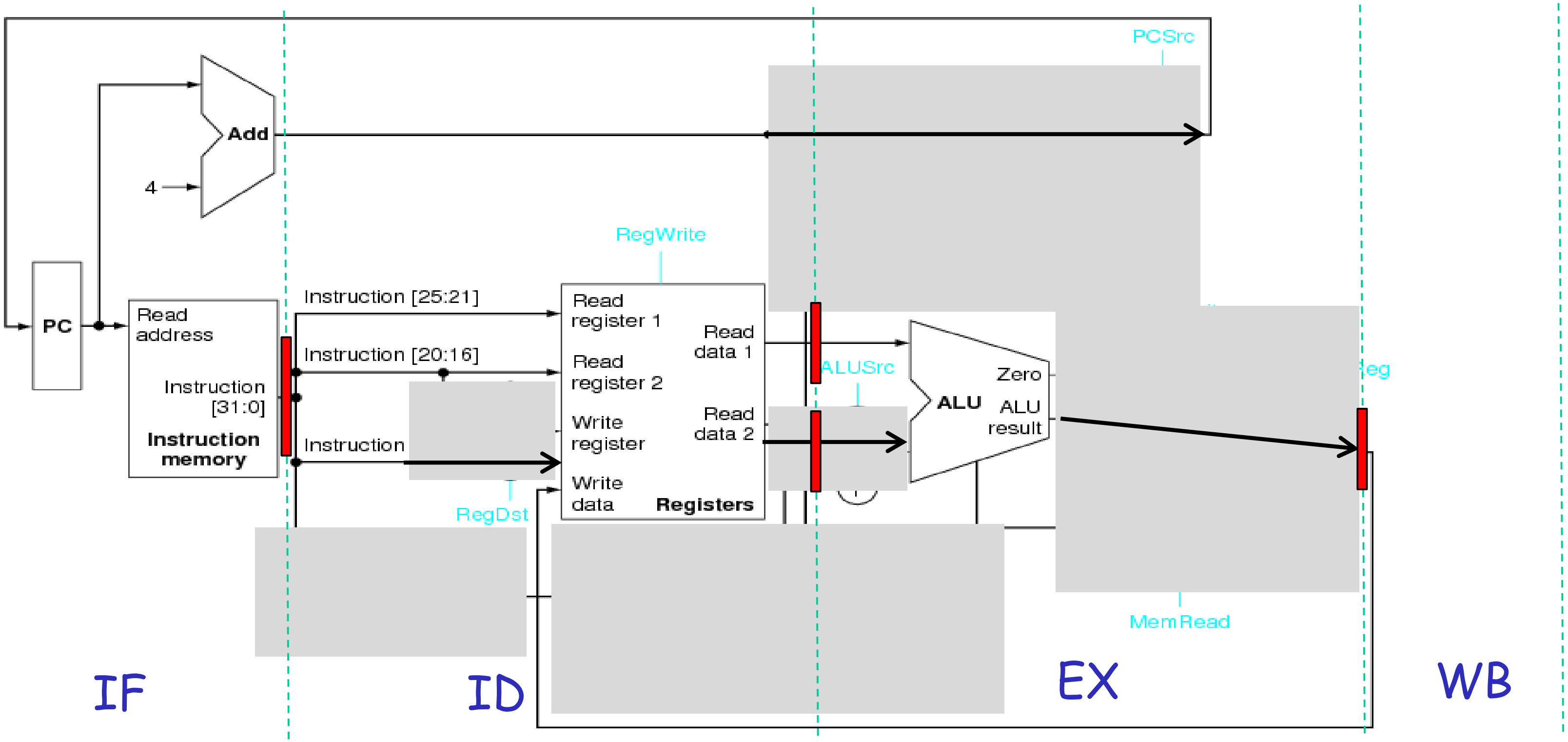
Towards four stage pipelined one supporting ADD

- IF: Instruction Fetch from instruction memory
- ID: Instruction Decode and operand fetch from regfile
- EX: EXecute operation
- WB: Write result Back to regfile



this slide is to be used as a whiteboard

The key : pipeline registers



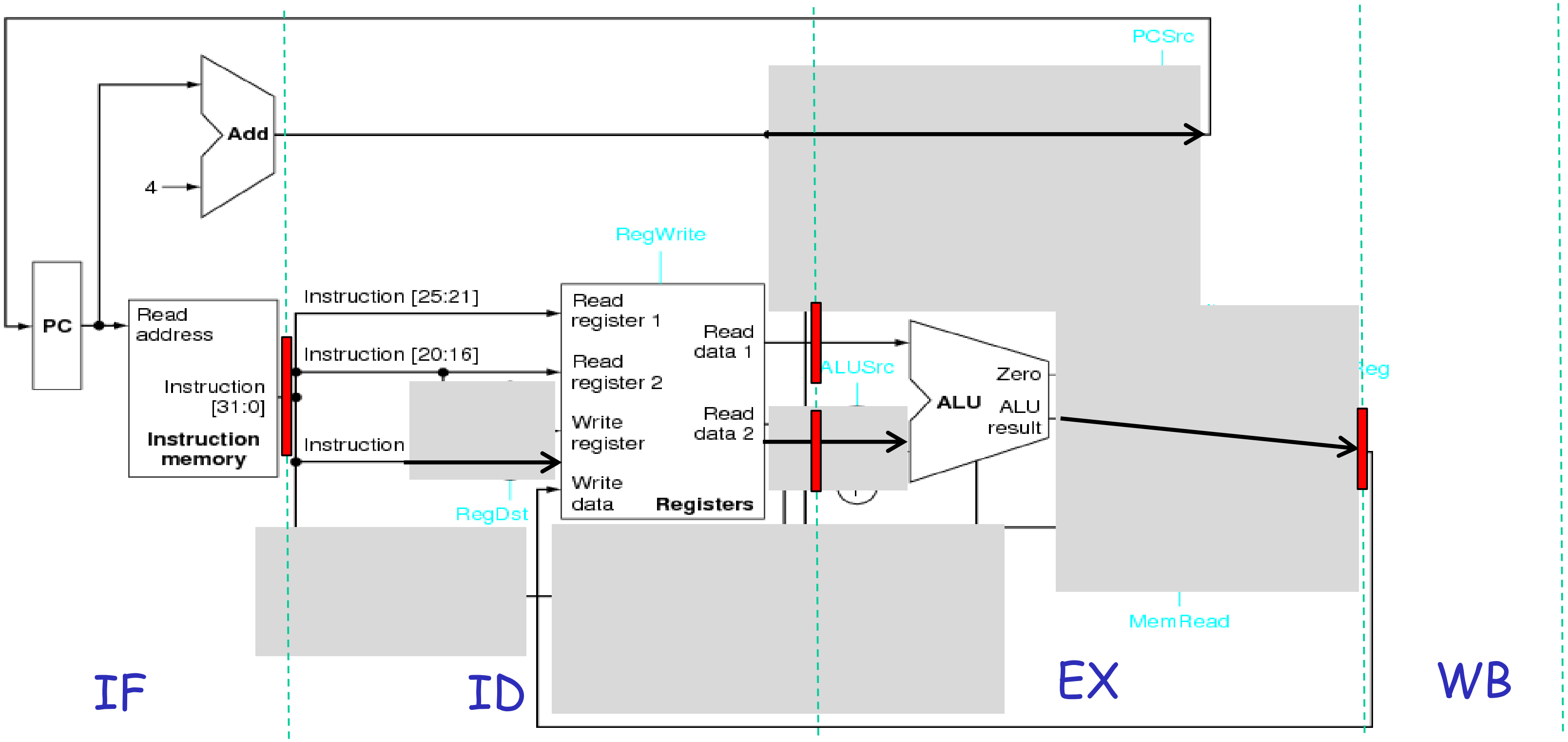
This datapath may have some errors and lackings.

Execution behavior of a pipelining processor

```
•   [1] 0x00: add  $0,  $0,  $0   # NOP, $0 <= 0 + 0
•   [2] 0x04: add  $1,  $1,  $1   # $1 <= 22 + 22
•   [3] 0x08: add  $2,  $2,  $2   # $2 <= 33 + 33
•   [4] 0x0c: add  $0,  $0,  $0   # NOP
•   [5] 0x10: add  $0,  $0,  $0   # NOP
•   [6] 0x14: add  $0,  $0,  $0   # NOP
```

assuming that the initial values of r[1]=22 and r[2]=33

cc1



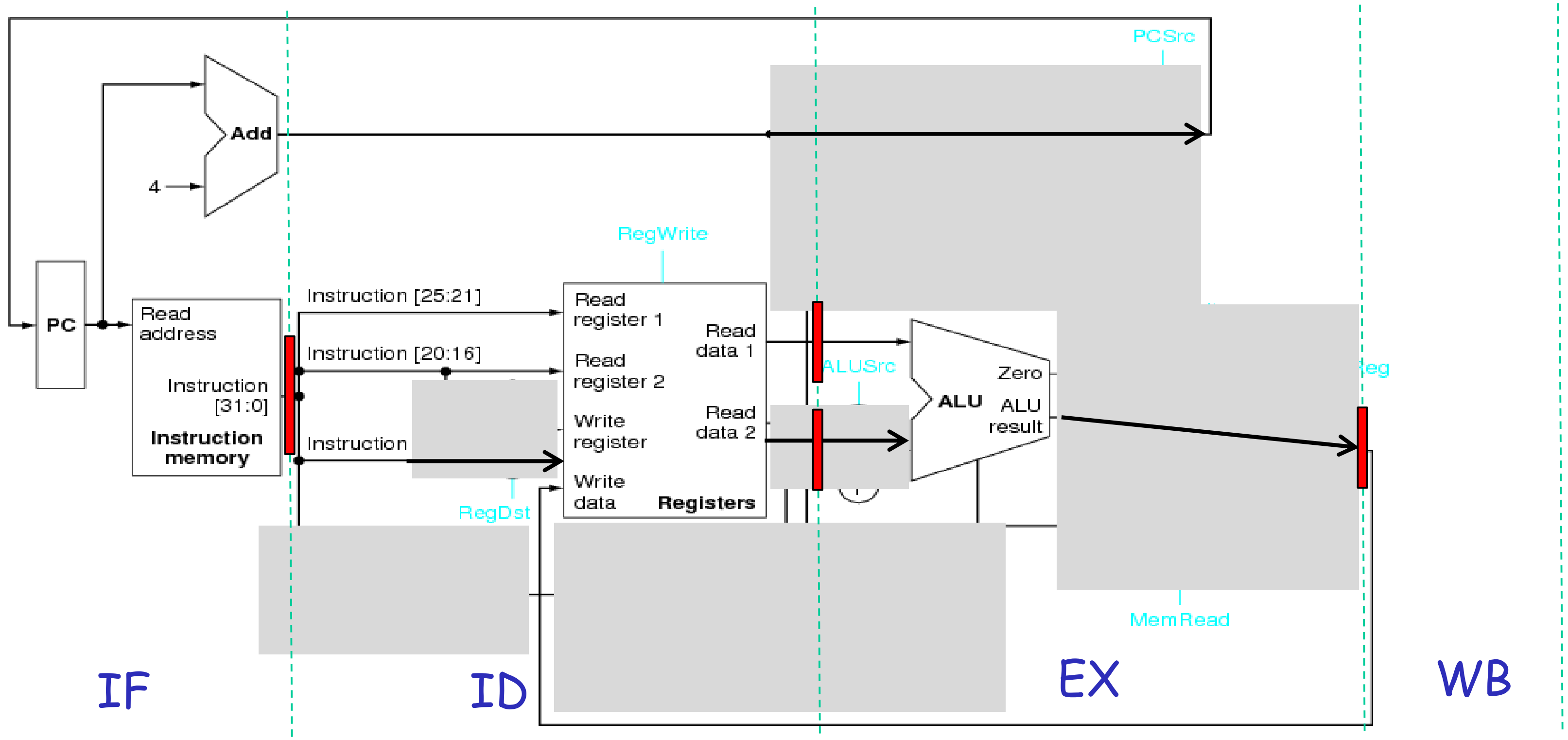
This datapath may have some errors and lackings.

Execution behavior of a pipelining processor

```
[1] 0x00: add  $0,  $0,  $0   # NOP, $0 <= 0 + 0
[2] 0x04: add  $1,  $1,  $1   # $1 <= 22 + 22
[3] 0x08: add  $2,  $2,  $2   # $2 <= 33 + 33
[4] 0x0c: add  $0,  $0,  $0   # NOP
[5] 0x10: add  $0,  $0,  $0   # NOP
[6] 0x14: add  $0,  $0,  $0   # NOP
```

assuming that the initial values of r[1]=22 and r[2]=33

cc2



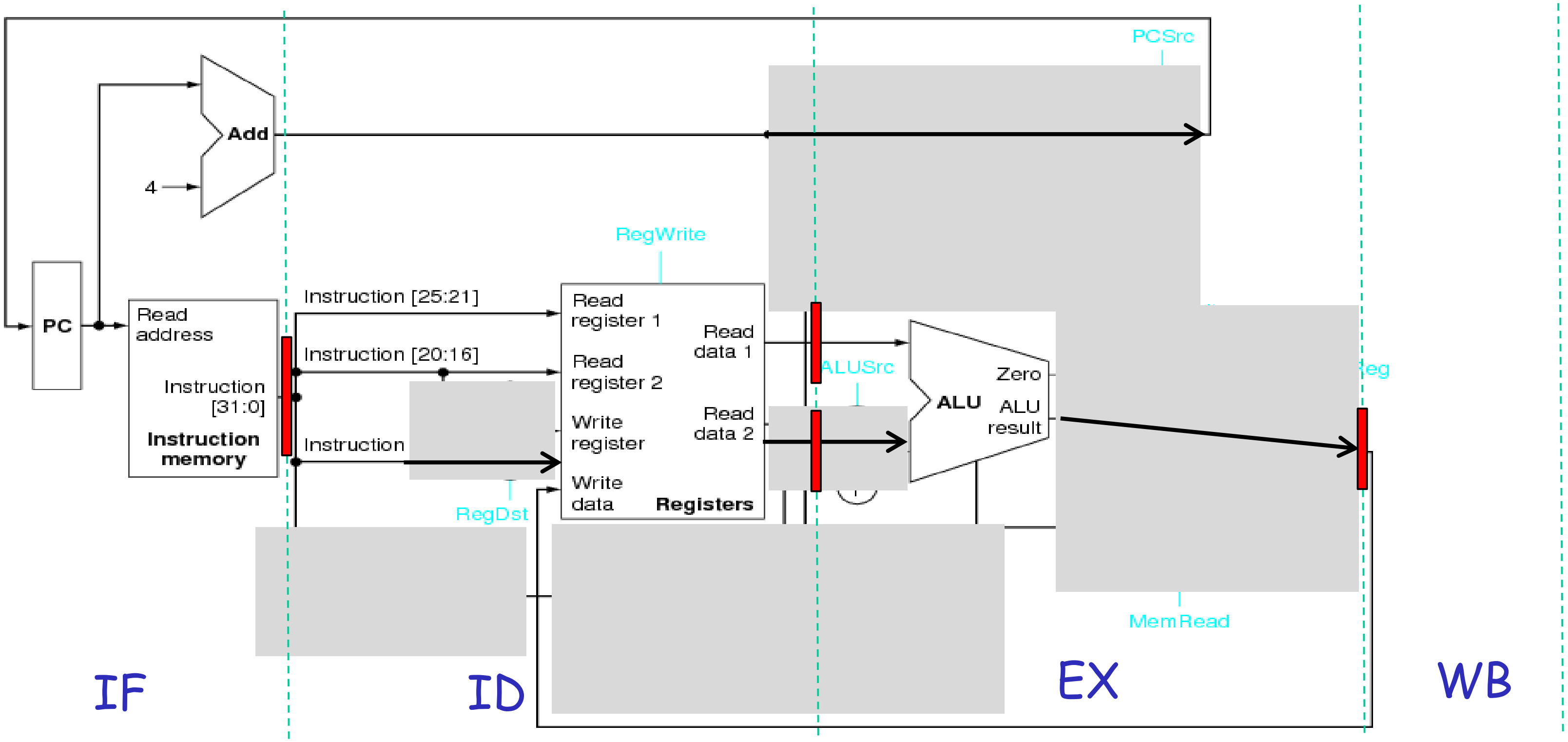
This datapath may have some errors and lackings.

Execution behavior of a pipelining processor

```
[1] 0x00: add  $0,  $0,  $0   # NOP, $0 <= 0 + 0
[2] 0x04: add  $1,  $1,  $1   # $1 <= 22 + 22
[3] 0x08: add  $2,  $2,  $2   # $2 <= 33 + 33
[4] 0x0c: add  $0,  $0,  $0   # NOP
[5] 0x10: add  $0,  $0,  $0   # NOP
[6] 0x14: add  $0,  $0,  $0   # NOP
```

assuming that the initial values of r[1]=22 and r[2]=33

cc3



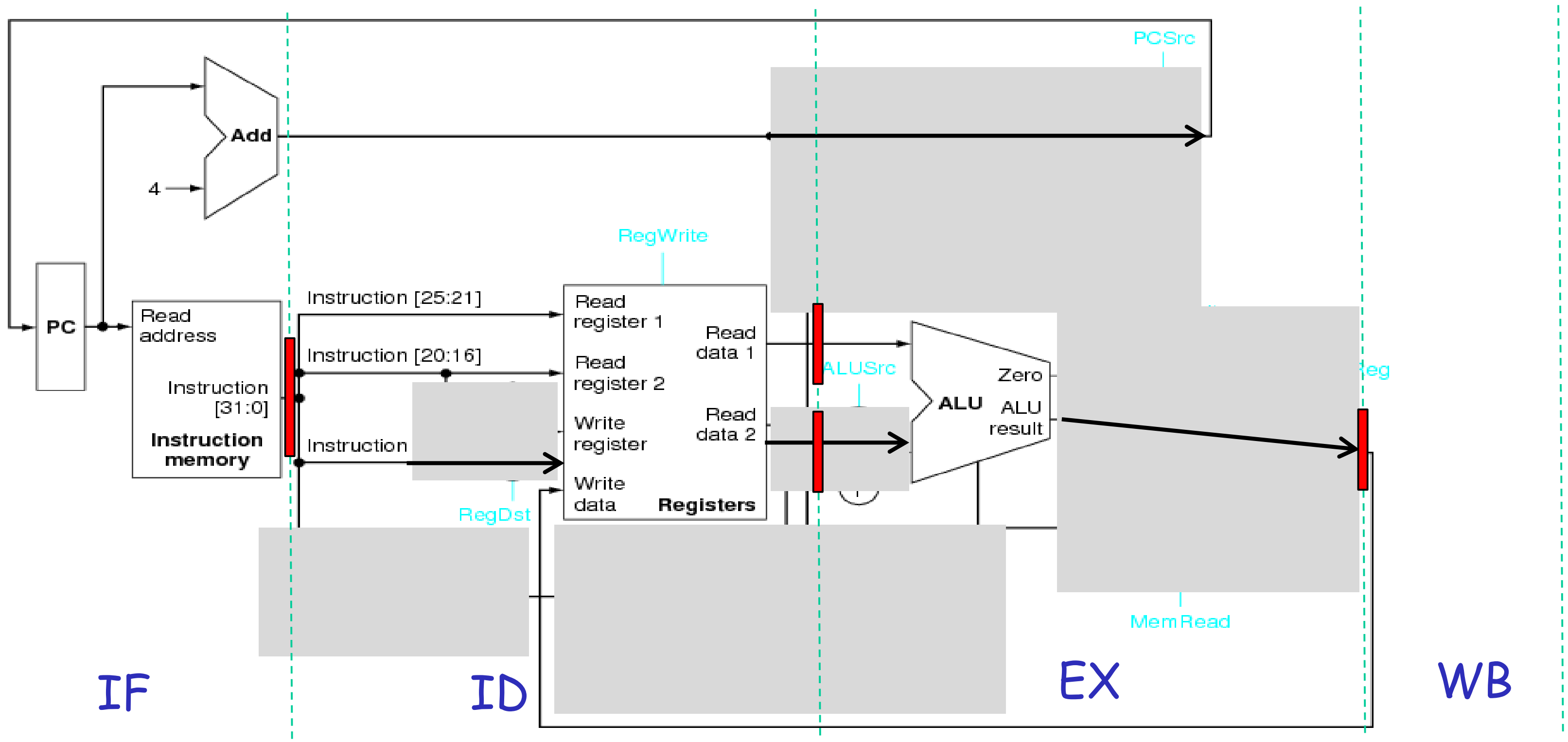
This datapath may have some errors and lackings.

Execution behavior of a pipelining processor

```
•    [1] 0x00: add  $0,  $0,  $0   # NOP, $0 <= 0 + 0
•    [2] 0x04: add  $1,  $1,  $1   # $1 <= 22 + 22
•    [3] 0x08: add  $2,  $2,  $2   # $2 <= 33 + 33
•    [4] 0x0c: add  $0,  $0,  $0   # NOP
•    [5] 0x10: add  $0,  $0,  $0   # NOP
•    [6] 0x14: add  $0,  $0,  $0   # NOP
```

assuming that the initial values of r[1]=22 and r[2]=33

cc4



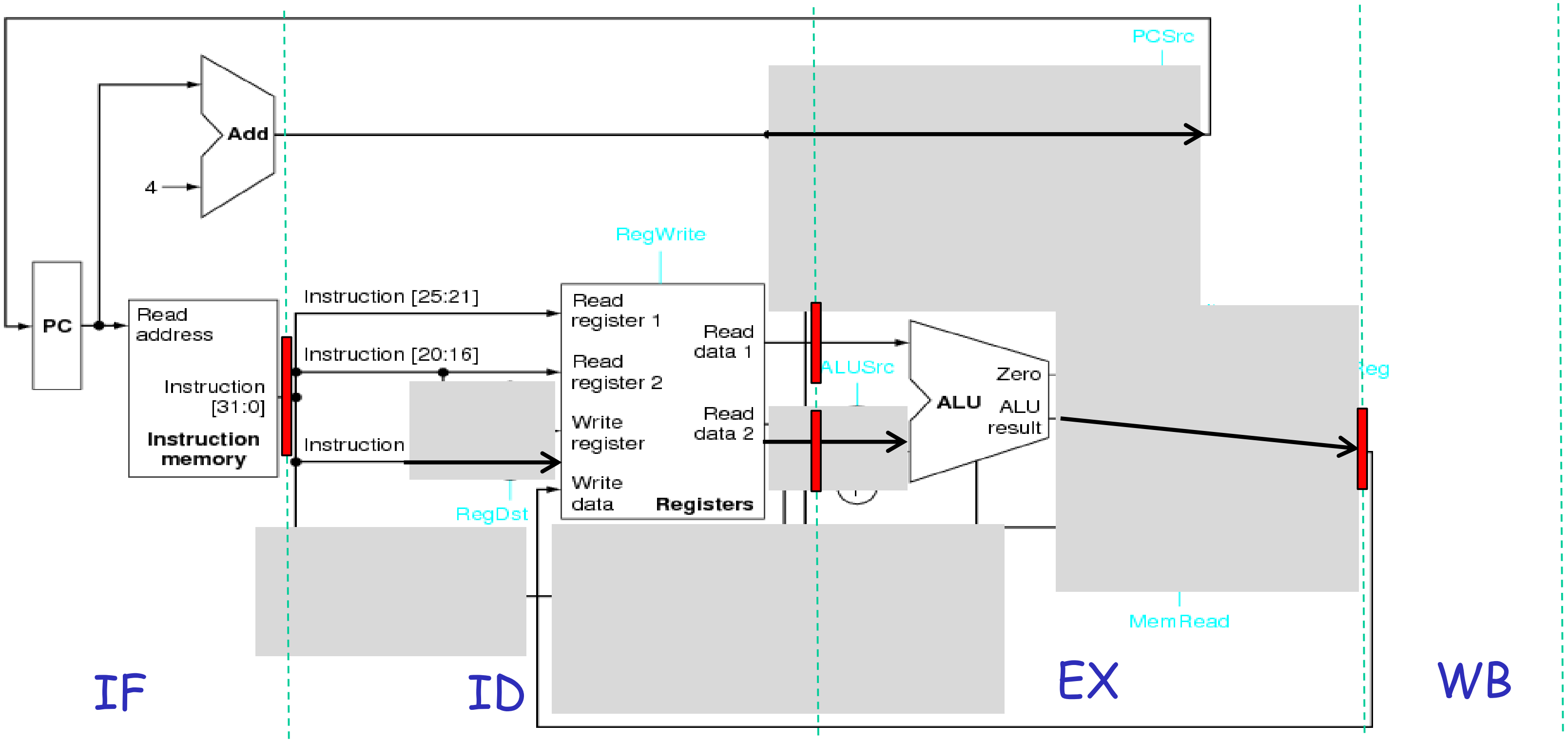
This datapath may have some errors and lackings.

Execution behavior of a pipelining processor

```
•   [1] 0x00: add  $0,  $0,  $0   # NOP, $0 <= 0 + 0
•   [2] 0x04: add  $1,  $1,  $1   # $1 <= 22 + 22
•   [3] 0x08: add  $2,  $2,  $2   # $2 <= 33 + 33
•   [4] 0x0c: add  $0,  $0,  $0   # NOP
•   [5] 0x10: add  $0,  $0,  $0   # NOP
•   [6] 0x14: add  $0,  $0,  $0   # NOP
```

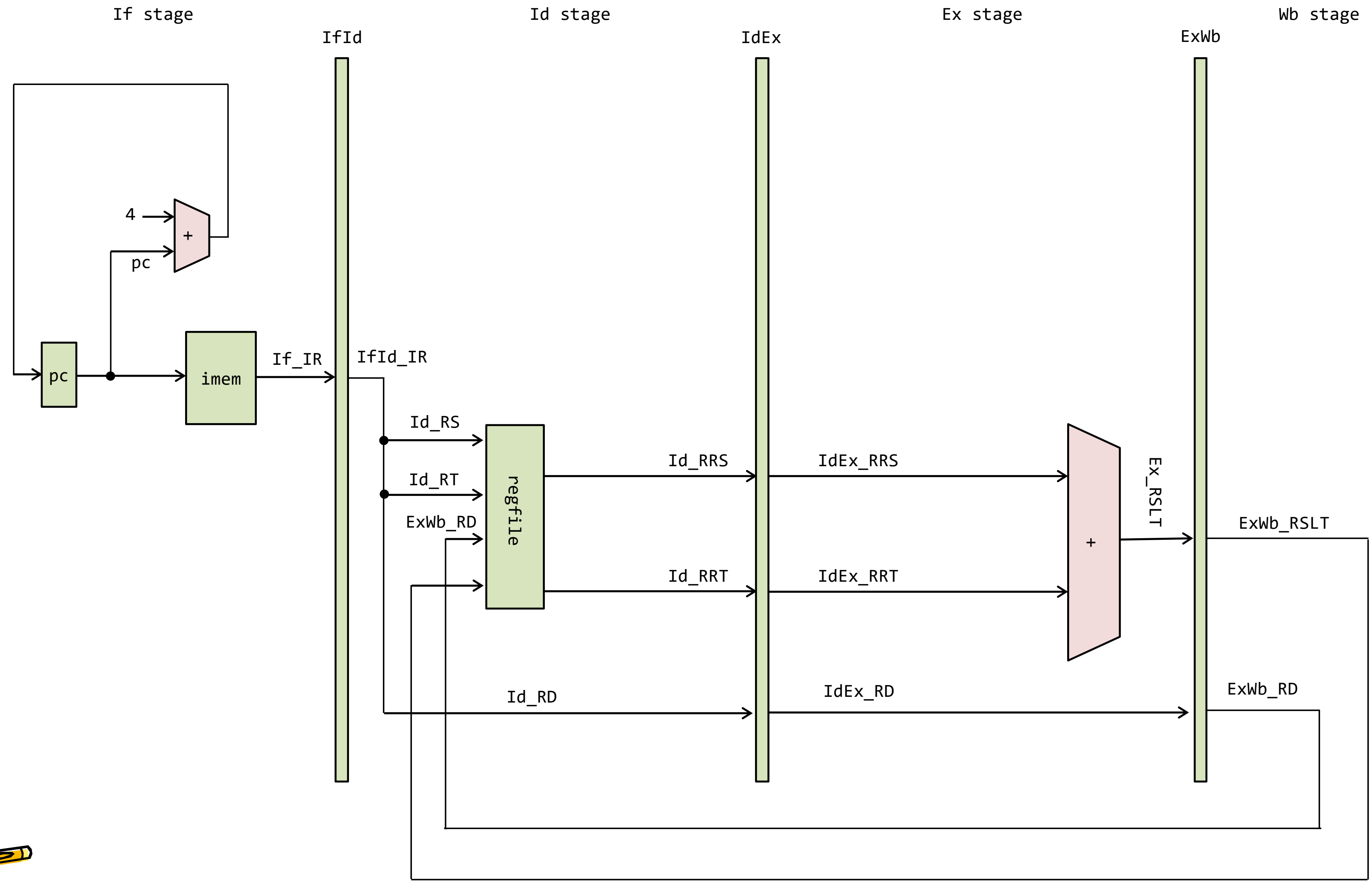
assuming that the initial values of r[1]=22 and r[2]=33

cc5



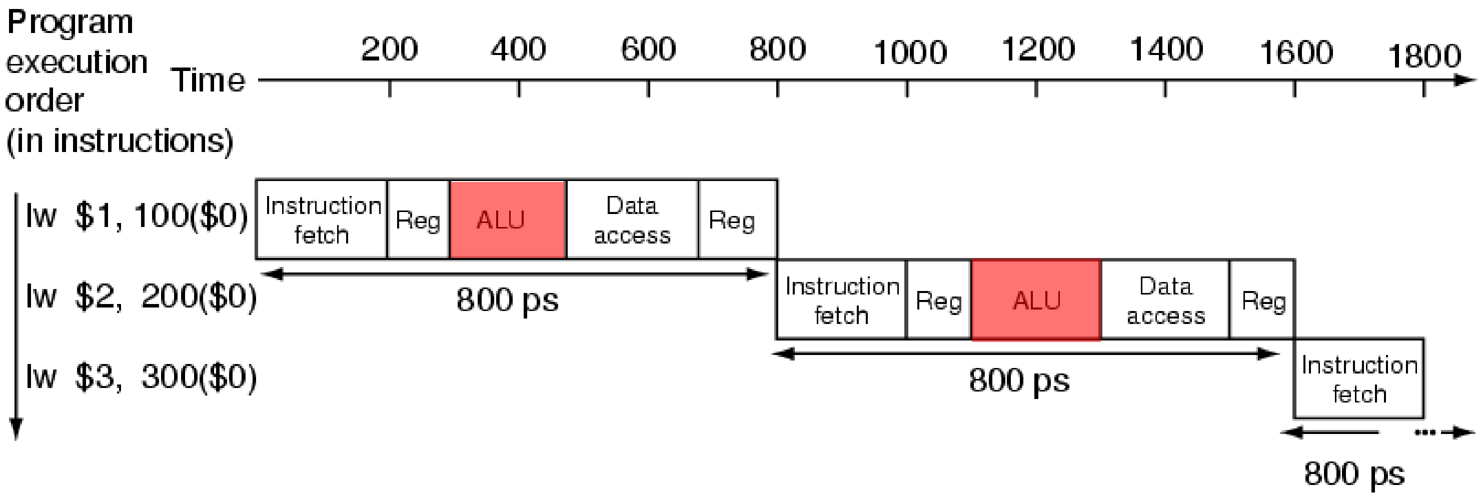
This datapath may have some errors and lackings.

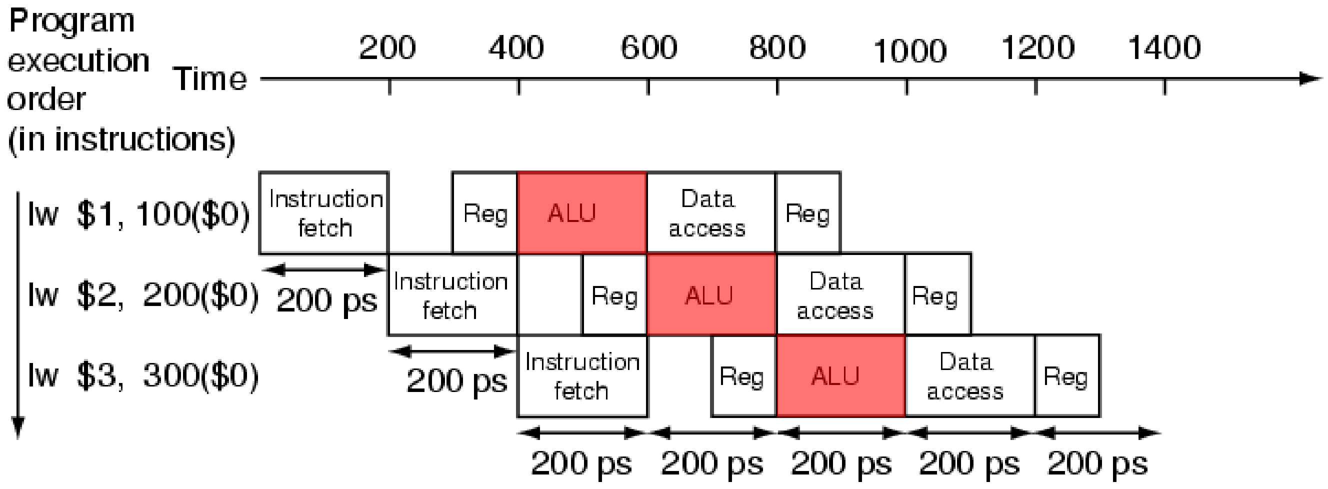
Four stage pipelined processor supporting ADD



this slide is to be used as a whiteboard

Single-cycle and pipelined processors





this slide is to be used as a whiteboard

Fiscal Year 2022

Ver. 2022-12-19a

Course number: CSC.T433
School of Computing,
Graduate major in Computer Science

Advanced Computer Architecture

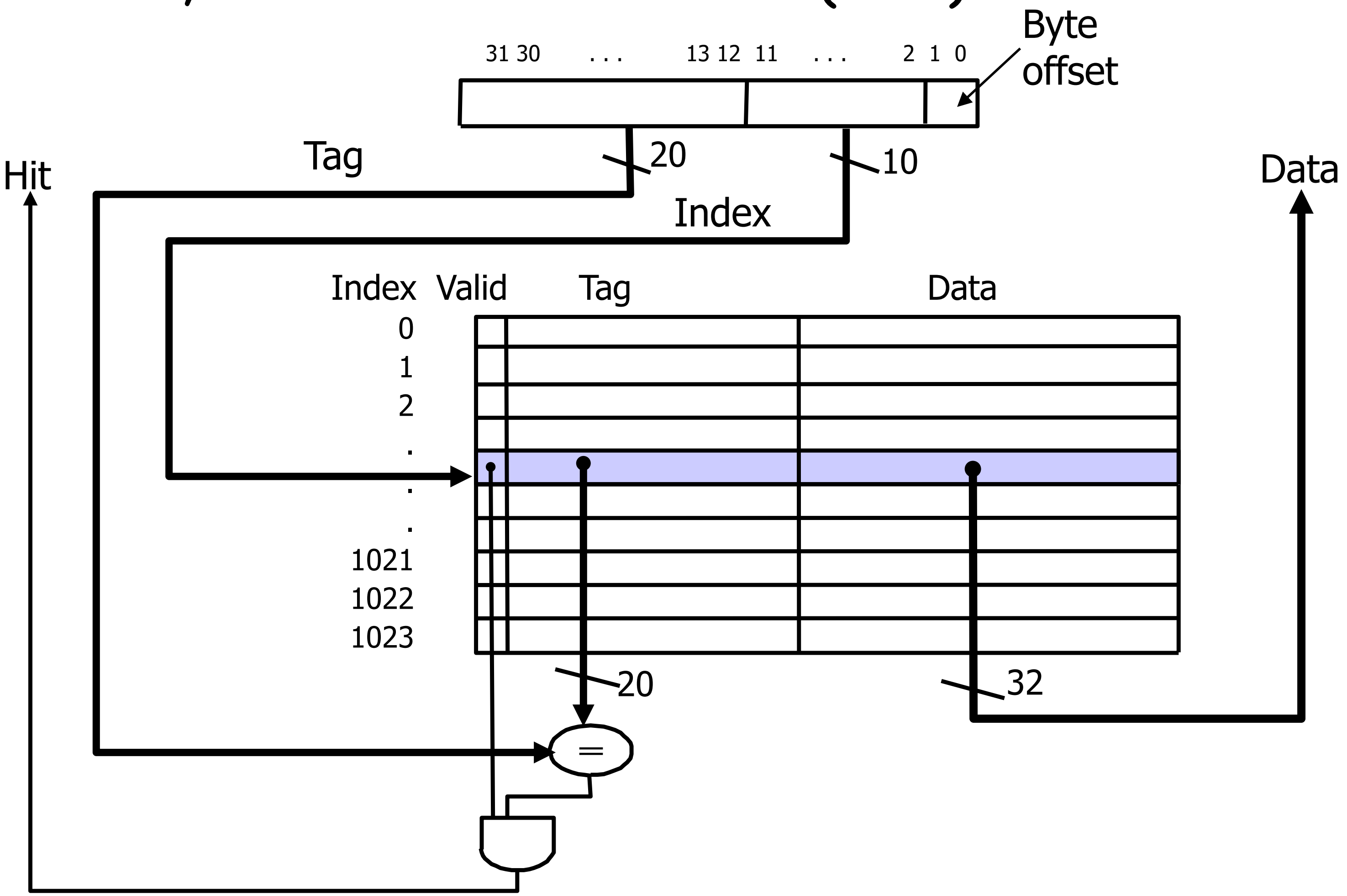
3. HDL, Single-cycle processor, and Memory Hierarchy Design

www.arch.cs.titech.ac.jp/lecture/ACA/
Room No.W831, HyFlex
Mon 13:45-15:25, Thr 13:45-15:25

Kenji Kise, Department of Computer Science
kise _at_ c.titech.ac.jp

MIPS Direct Mapped Cache Example

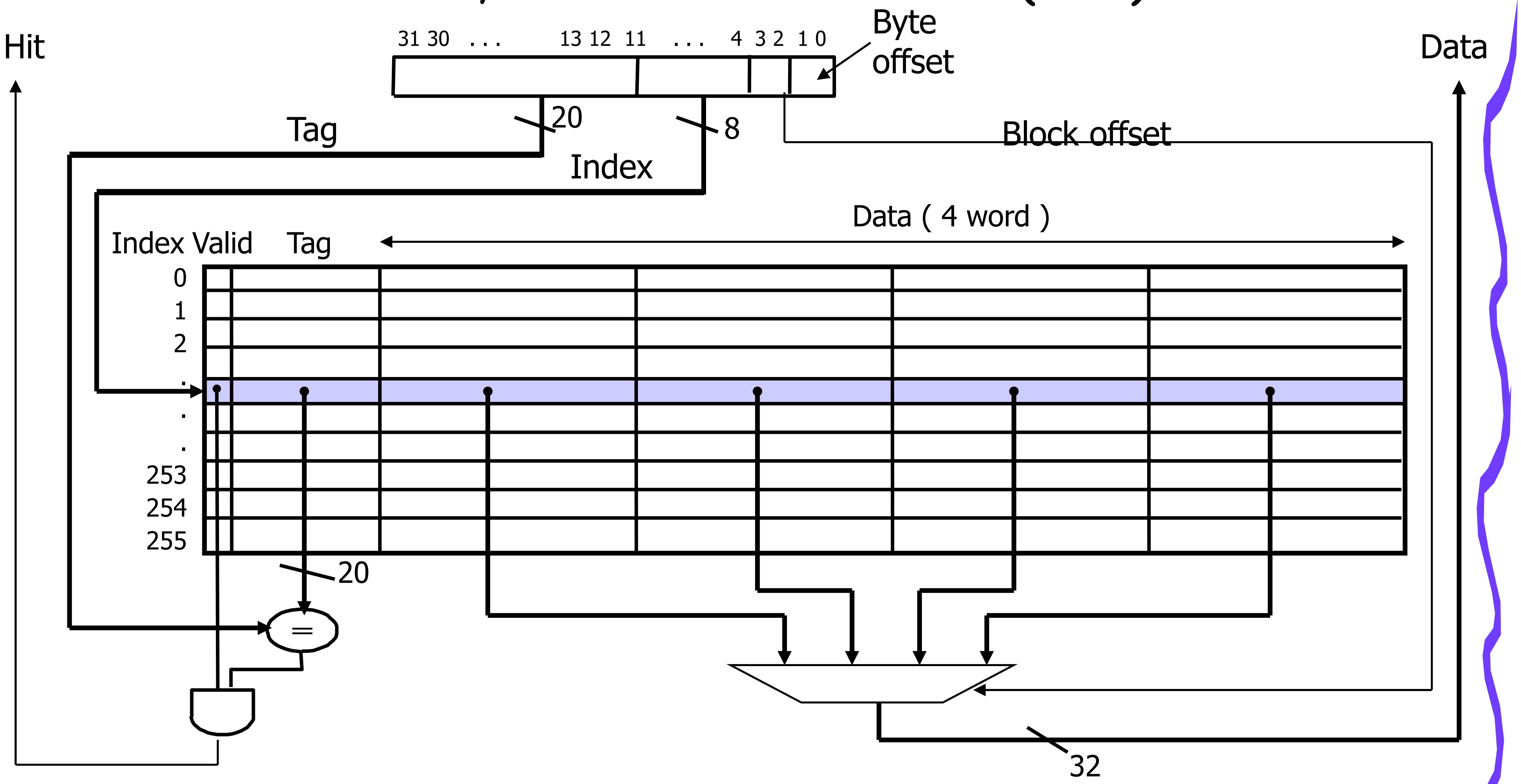
- One word/block, cache size = 1K words (4KB)



What kind of locality are we taking advantage of?

Multiword Block Direct Mapped Cache

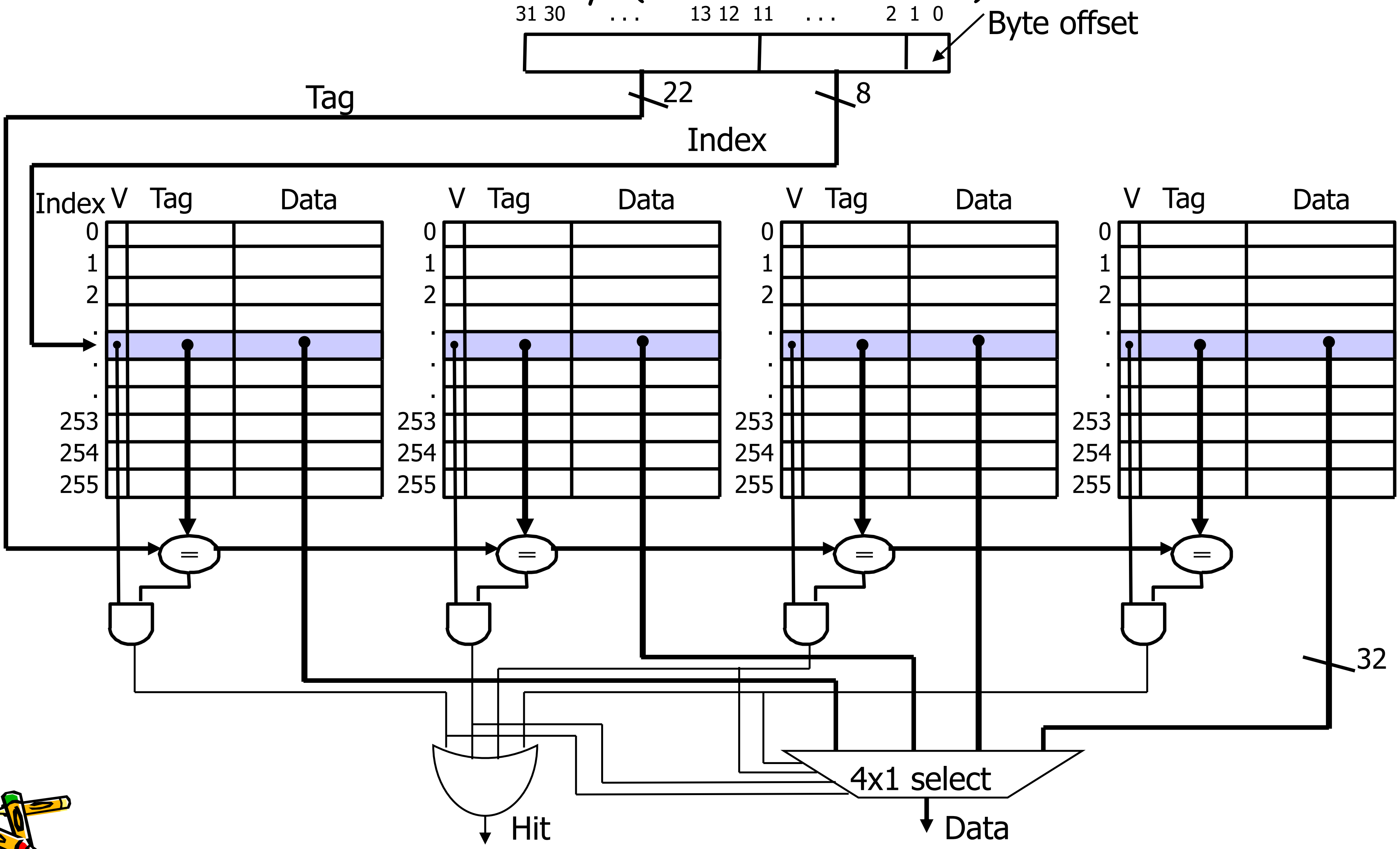
- Four words/block, cache size = 1K words (4KB)



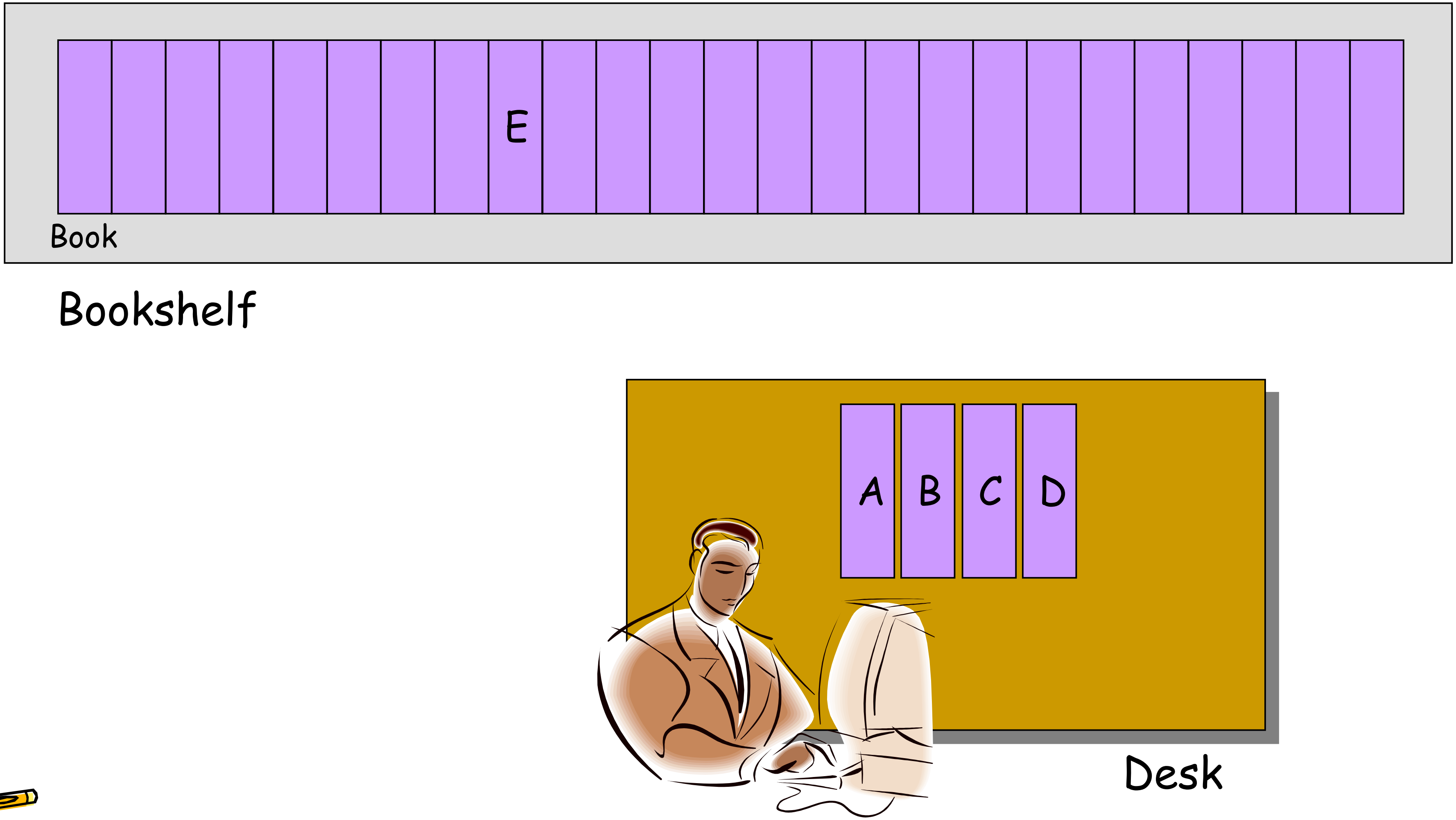
What kind of locality are we taking advantage of?

Four-Way Set Associative Cache

- 2^8 = **256 sets** each with four ways (each with one block)



Cache Associativity & Replacement Policy



Costs of Set Associative Caches

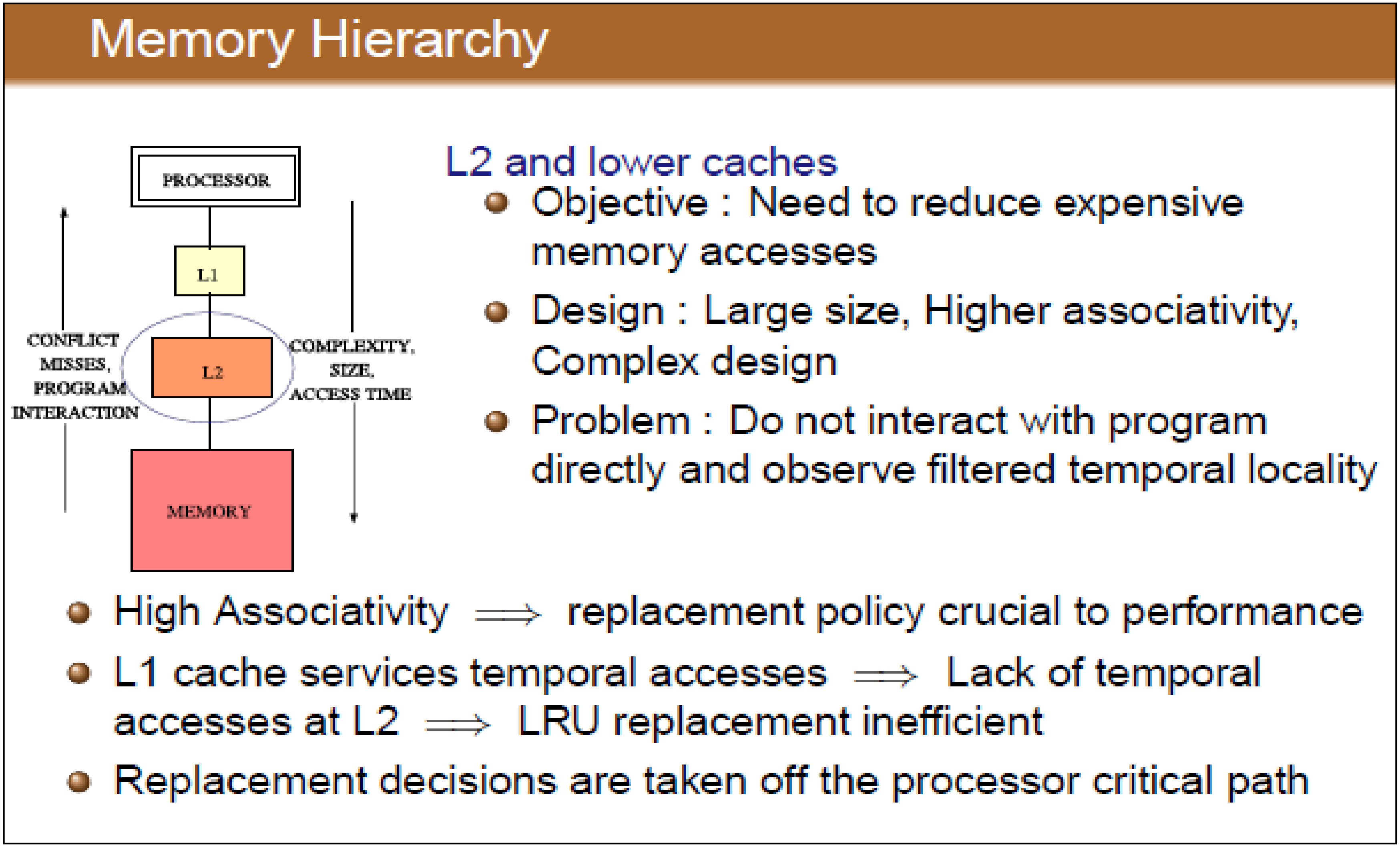
- N-way set associative cache costs
  - N comparators (delay and area)
  - MUX delay (set selection) before data is available
  - Data available after set selection and Hit/Miss decision.
- When a miss occurs,
  which way's block do we pick for replacement ?
  - Least Recently Used (LRU):
    the block replaced is the one that has been unused for the longest time
    - Must have hardware to keep track of when each way's block was used
    - For 2-way set associative, takes one bit per set →
      set the bit when a block is referenced
      (and reset the other way's bit)
  - Random

Recommended Reading

- Emulating Optimal Replacement with a Shepherd Cache
  - Kaushik Rajan, Govindarajan Ramaswamy, Indian Institute of Science
  - MICRO-40, pp. 445-454, 2007
  - Session 8: Cache Replacement Policies

- A quote:
  "The inherent temporal locality in memory accesses is filtered out by the L1 cache. As a consequence, an L2 cache with LRU replacement incurs significantly higher misses than the optimal replacement policy (OPT). We propose to narrow this gap through a novel replacement strategy that mimics the replacement decisions of OPT."

Memory Hierarchy Design



Memory Hierarchy

PROCESSOR
L1
L2
MEMORY
CONFLICT MISSES, PROGRAM INTERACTION
COMPLEXITY, SIZE, ACCESS TIME

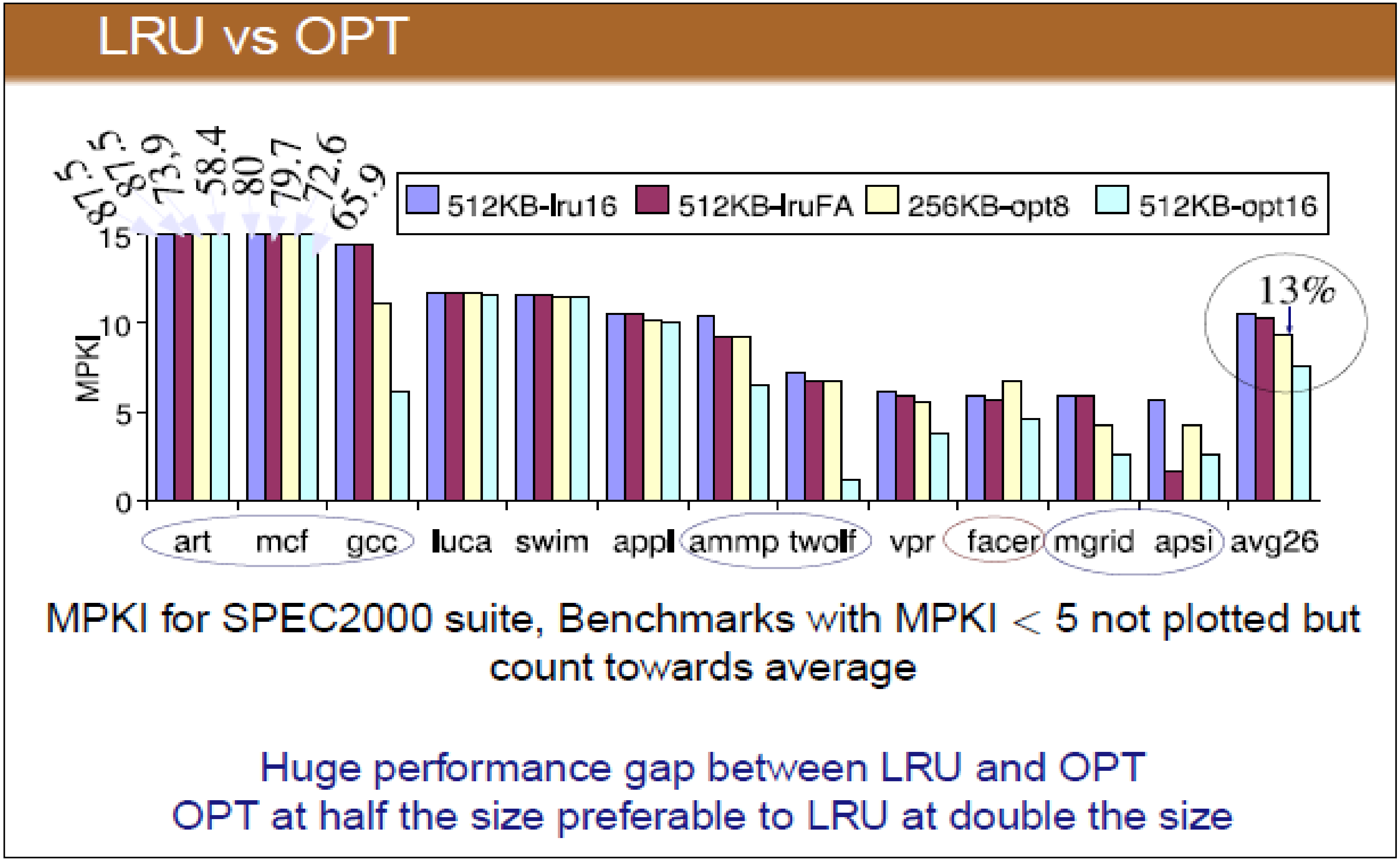
L2 and lower caches

- Objective : Need to reduce expensive memory accesses
- Design : Large size, Higher associativity, Complex design
- Problem : Do not interact with program directly and observe filtered temporal locality

- High Associativity ⟹ replacement policy crucial to performance
- L1 cache services temporal accesses ⟹ Lack of temporal accesses at L2 ⟹ LRU replacement inefficient
- Replacement decisions are taken off the processor critical path

Emulating Optimal Replacement with a Shepherd Cache, MICRO-2007

LRU has room for improvement



LRU vs OPT

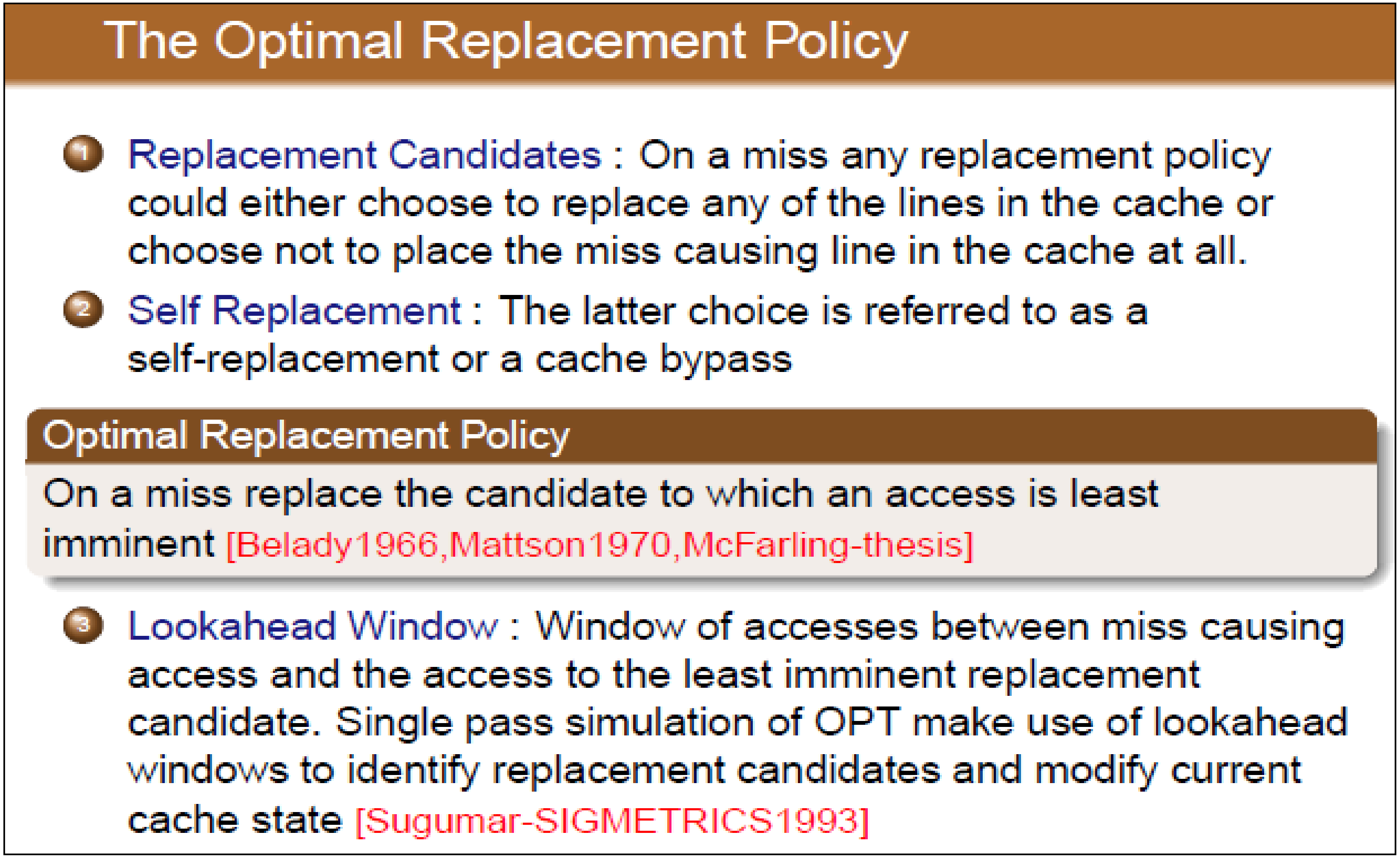
MPKI for SPEC2000 suite, Benchmarks with MPKI < 5 not plotted but count towards average

Huge performance gap between LRU and OPT
OPT at half the size preferable to LRU at double the size

MPKI: Miss Per Kilo Instructions

Emulating Optimal Replacement with a Shepherd Cache, MICRO-2007

OPT: Optimal Replacement Policy



The Optimal Replacement Policy

1. Replacement Candidates : On a miss any replacement policy could either choose to replace any of the lines in the cache or choose not to place the miss causing line in the cache at all.
2. Self Replacement : The latter choice is referred to as a self-replacement or a cache bypass

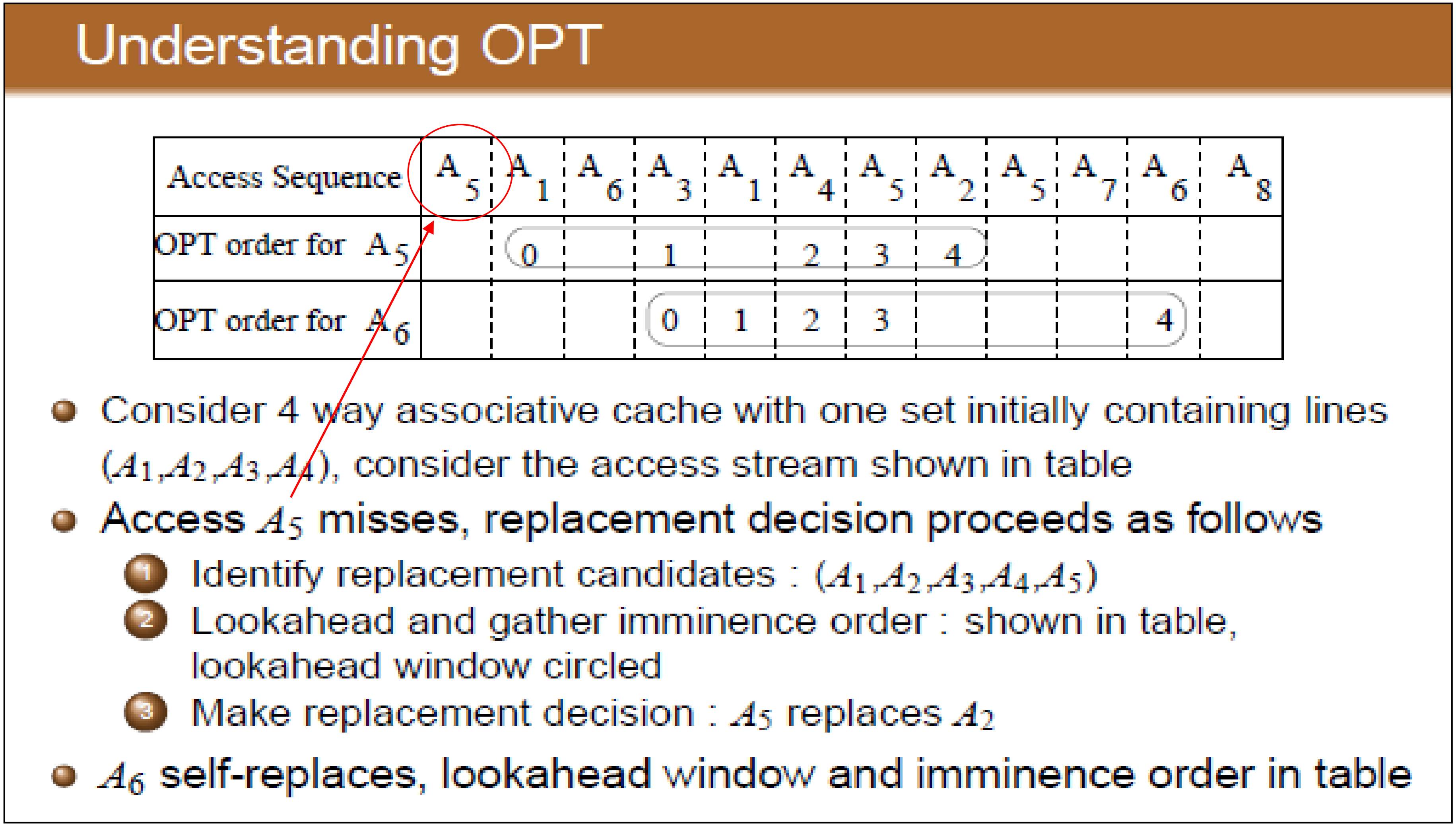
Optimal Replacement Policy

On a miss replace the candidate to which an access is least imminent [Belady1966,Mattson1970,McFarling-thesis]

3. Lookahead Window : Window of accesses between miss causing access and the access to the least imminent replacement candidate. Single pass simulation of OPT make use of lookahead windows to identify replacement candidates and modify current cache state [Sugumar-SIGMETRICS1993]

Emulating Optimal Replacement with a Shepherd Cache, MICRO-2007

Example of Optimal Replacement Policy



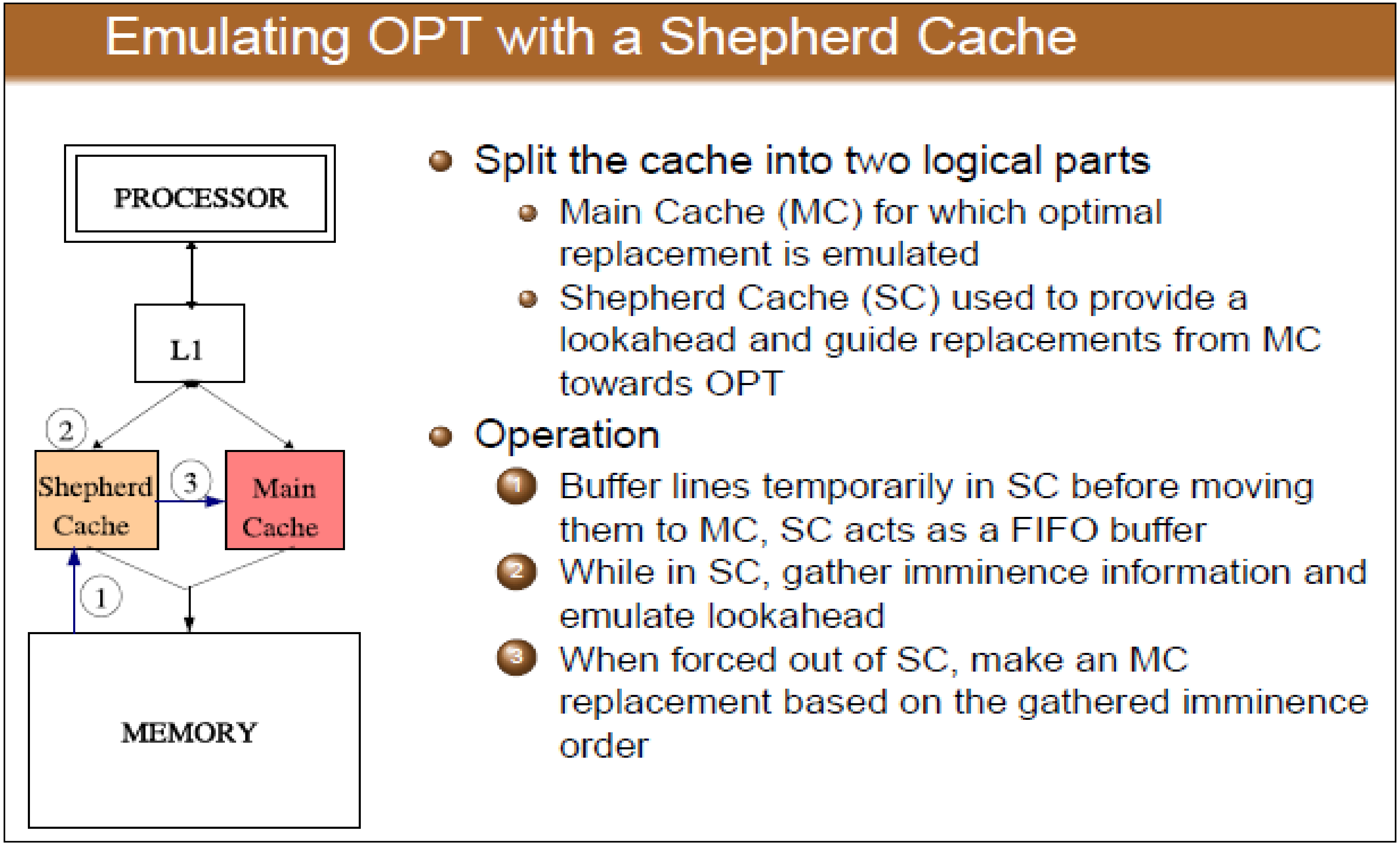
Understanding OPT

Access Sequence	A_5	A_1	A_6	A_3	A_1	A_4	A_5	A_2	A_5	A_7	A_6	A_8
OPT order for A_5		0		1		2	3	4				
OPT order for A_6				0	1	2	3				4	

- Consider 4 way associative cache with one set initially containing lines (A_1,A_2,A_3,A_4), consider the access stream shown in table
- Access A_5 misses, replacement decision proceeds as follows
  1. Identify replacement candidates : (A_1,A_2,A_3,A_4,A_5)
  2. Lookahead and gather imminence order : shown in table, lookahead window circled
  3. Make replacement decision : A_5 replaces A_2
- A_6 self-replaces, lookahead window and imminence order in table

Emulating Optimal Replacement with a Shepherd Cache, MICRO-2007

Shepherd Cache emulation OPT

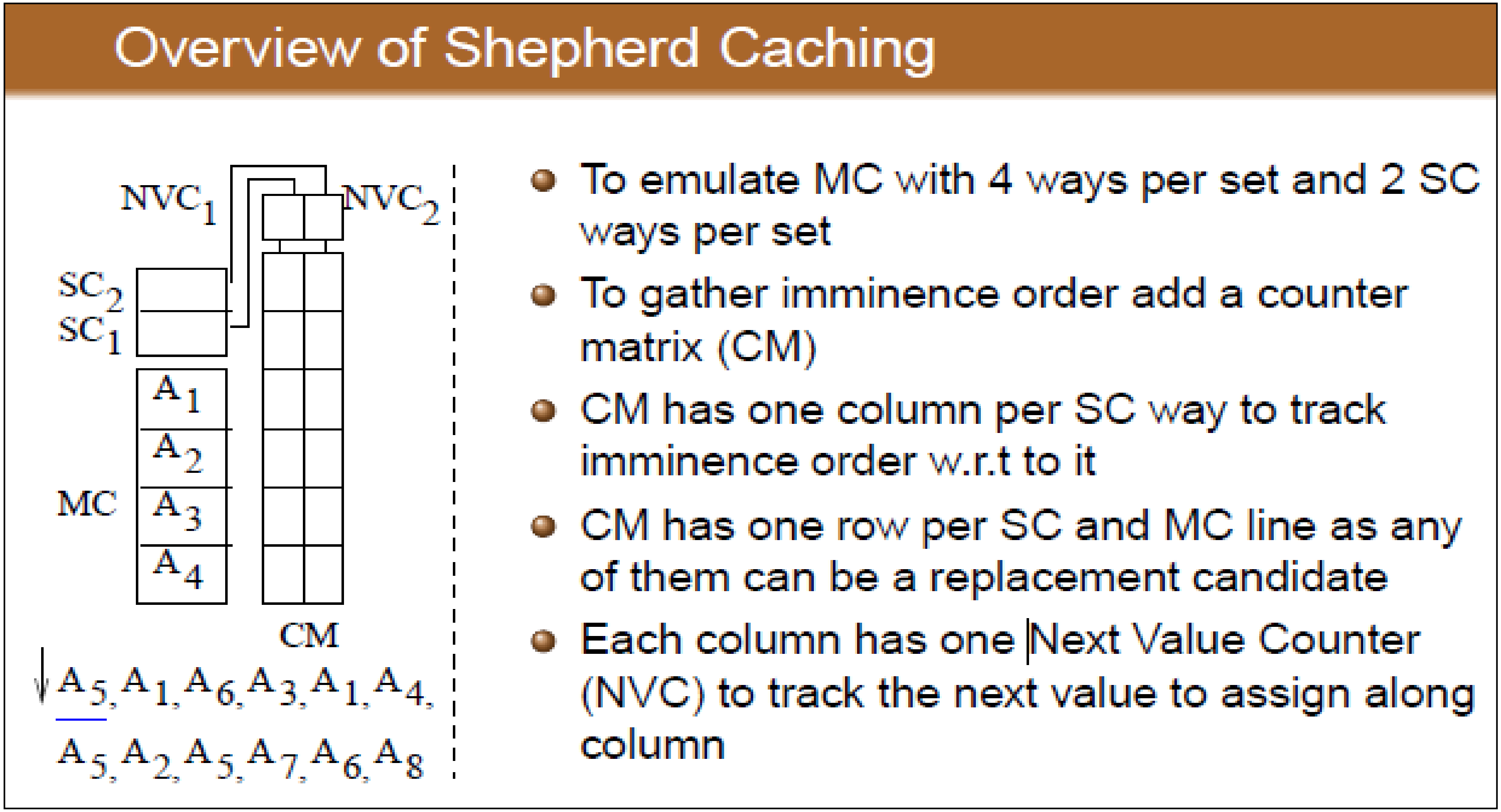


Emulating OPT with a Shepherd Cache

- Split the cache into two logical parts
  - Main Cache (MC) for which optimal replacement is emulated
  - Shepherd Cache (SC) used to provide a lookahead and guide replacements from MC towards OPT
- Operation
  1. Buffer lines temporarily in SC before moving them to MC, SC acts as a FIFO buffer
  2. While in SC, gather imminence information and emulate lookahead
  3. When forced out of SC, make an MC replacement based on the gathered imminence order

Emulating Optimal Replacement with a Shepherd Cache, MICRO-2007

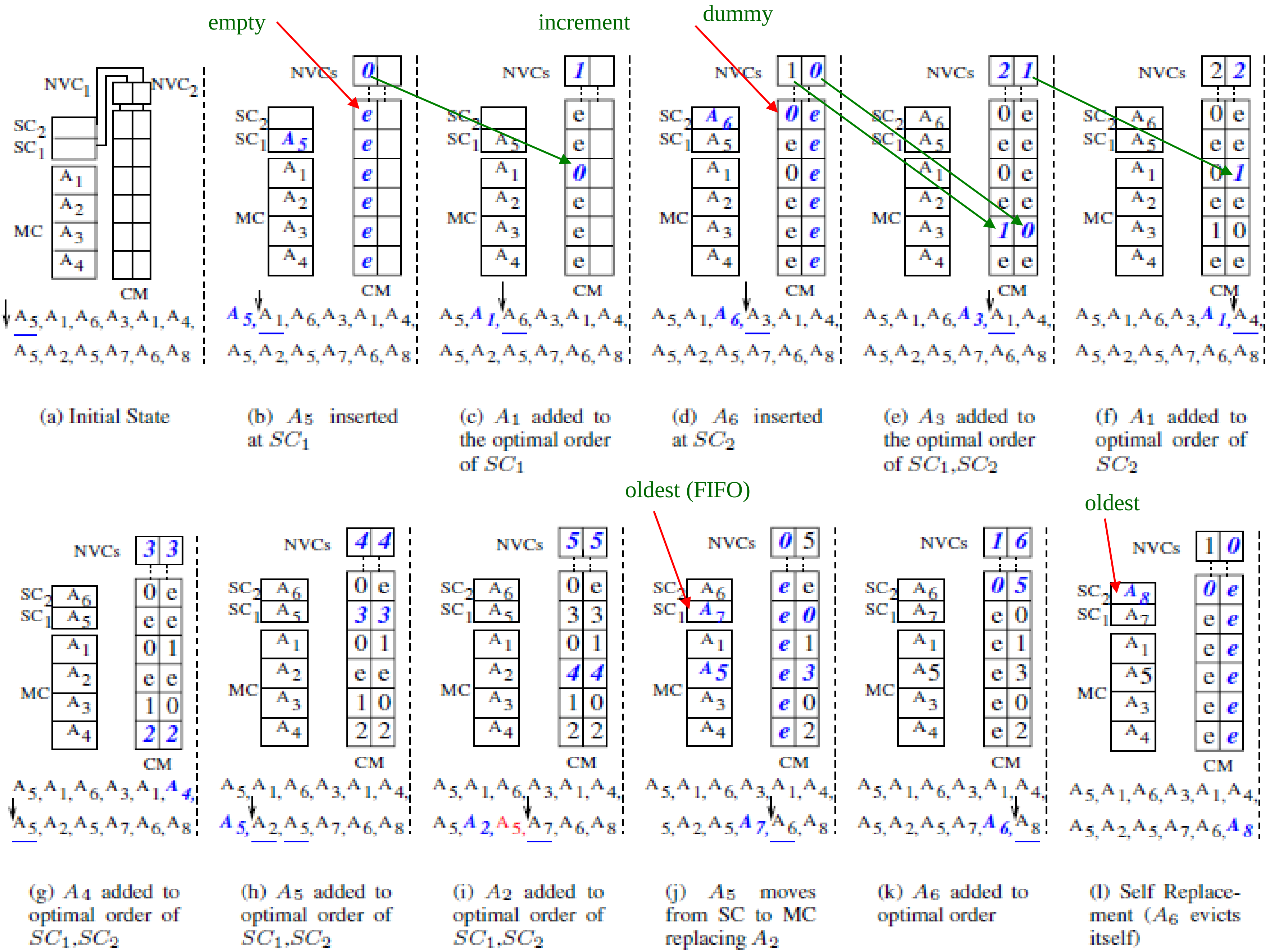
Shepherd Cache Overview



Overview of Shepherd Caching

- To emulate MC with 4 ways per set and 2 SC ways per set
- To gather imminence order add a counter matrix (CM)
- CM has one column per SC way to track imminence order w.r.t to it
- CM has one row per SC and MC line as any of them can be a replacement candidate
- Each column has one Next Value Counter (NVC) to track the next value to assign along column

Emulating Optimal Replacement with a Shepherd Cache, MICRO-2007



(a) Initial State

(b) A_5 inserted at SC_1

(c) A_1 added to the optimal order of SC_1

(d) A_6 inserted at SC_2

(e) A_3 added to the optimal order of SC_1,SC_2

(f) A_1 added to optimal order of SC_2

(g) A_4 added to optimal order of SC_1,SC_2

(h) A_5 added to optimal order of SC_1,SC_2

(i) A_2 added to optimal order of SC_1,SC_2

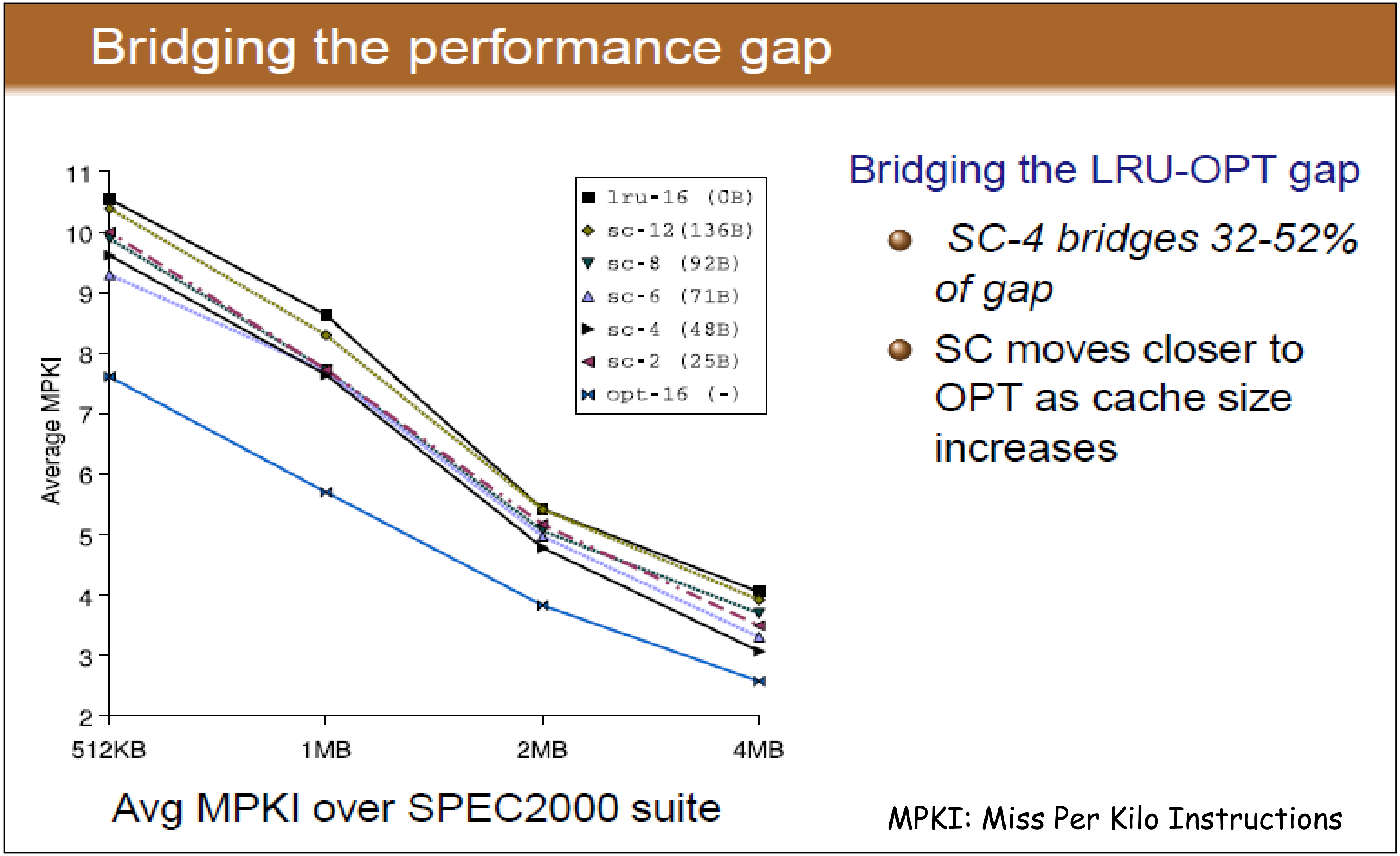
(j) A_5 moves from SC to MC replacing A_2

(k) A_6 added to optimal order

(l) Self Replacement (A_6 evicts itself)

Emulating Optimal Replacement with a Shepherd Cache, MICRO-2007

Shepherd cache bridges 32 - 52% of the gap



Bridging the performance gap

Avg MPKI over SPEC2000 suite

Bridging the LRU-OPT gap

- *SC-4 bridges 32-52% of gap*
- SC moves closer to OPT as cache size increases

MPKI: Miss Per Kilo Instructions

Emulating Optimal Replacement with a Shepherd Cache, MICRO-2007