cs704fig_ilp1.cdr
Sunday, March 14, 2021 7:50:31 PM
Color profile: Disabled

Composite Default screen

The Processor

➪ [Processor] architecture

In earlier machines the instruction set, addressing modes, register and memory arrangements closely matched the hardware.

The processor has 2 personalities depending on where you look!

➪ The “ active part ”

- ✎ Follow instructions
- ✎ Perform requested operations

Instruction set architecture (ISA) remains useful as a specification for writing machine software.

➪ Programmer interface

What functions are available for use

➪ Implementation details

How functions organized and performed

© 2021 Dr. Muhammad Al-Hashimi
COPYRIGHT 1998-2005 MORGAN KAUFMANN PUBLISHERS, INC. ALL RIGHTS RESERVED

cs704fig_ilp1.cdr
Sunday, March 14, 2021 7:50:31 PM
Color profile: Disabled

Composite Default screen

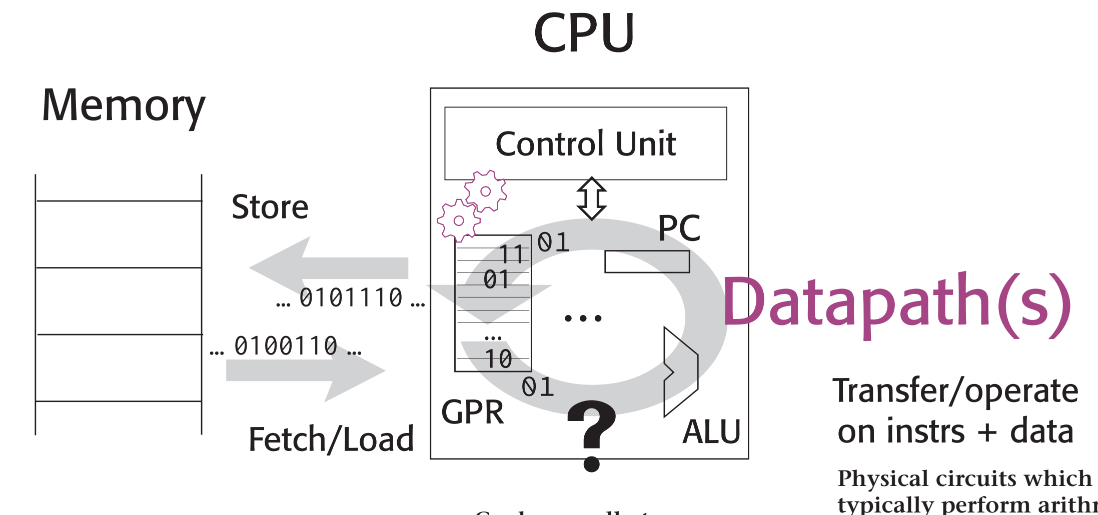
Processor Implementation
The Datapath

➪ Microarchitecture

In simplest scenario, a processor has to deal with a **stream** of instructions and data coming from different threads of many programs stored in memory.

At least a datapath for core instructions (default 2's comp operands, `int` in C-like langs), there may be more paths for specialized operands like FP or SIMD instructions.

Opcodes distinguish which operations to perform (hardware resources), how to obtain operands and which variants to use.



Physical circuits which typically perform arithmetic, logical, branch, and load-store operations.

Goal generally to minimize instruction time (**latency**) and maximize stream rate.

© 2021 Dr. Muhammad Al-Hashimi
COPYRIGHT 1998-2005 MORGAN KAUFMANN PUBLISHERS, INC. ALL RIGHTS RESERVED

cs704fig_ilp1.cdr
Sunday, March 14, 2021 7:50:31 PM
Color profile: Disabled

Composite Default screen

Instruction Implementation
Datapath Resources

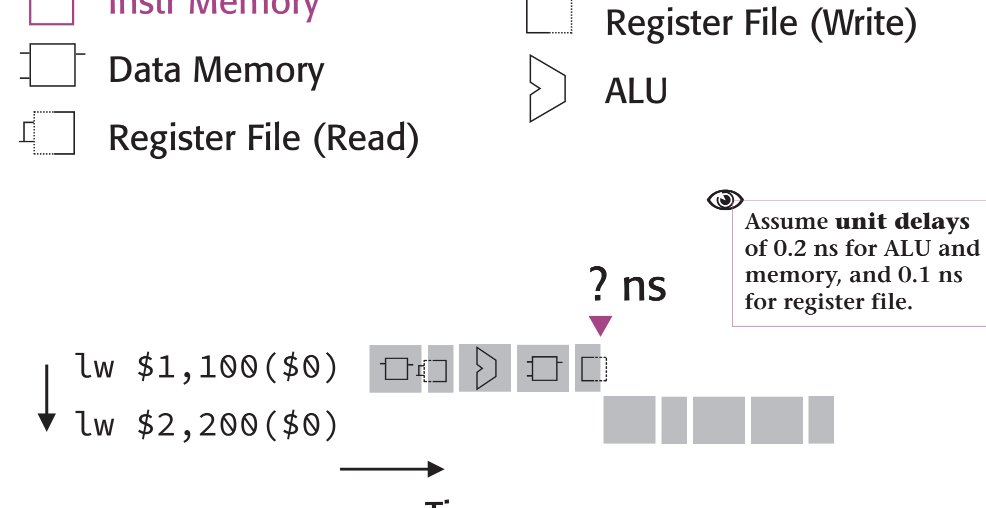
- Register file
- Architectural regs

Load word has to go through specified functions in sequence regardless of how a datapath is designed.

Example: Load Word

Specification (ISA)
- Fetch instr word from instr memory
- Load base addr from $0
- Sum base addr and constant passed in instr
- Fetch data word using sum as addr
- Store data word in $1 (or $2)

Required Functions
- Read an **Instr Memory**
- Read access a **Register File**
- Perform op in **ALU**
- Read a **Data Memory**
- Write access **Register File**



Assume **unit delays** of 0.2 ns for ALU and memory, and 0.1 ns for register file.

© 2021 Dr. Muhammad Al-Hashimi
COPYRIGHT 1998-2005 MORGAN KAUFMANN PUBLISHERS, INC. ALL RIGHTS RESERVED

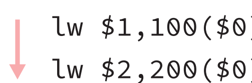
cs704fig_ilp1.cdr
Sunday, March 14, 2021 7:50:31 PM
Color profile: Disabled

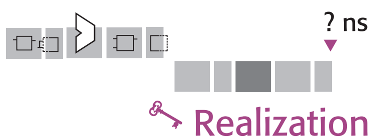
Composite Default screen

Instruction Overlapping

We can identify an execution **stage** characterized by which **datapath resources** are used.

➪ Sequential execution





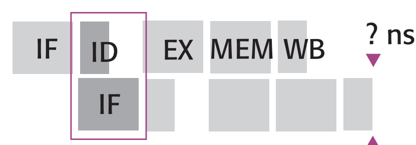
No 2 consecutive stages use the same units; in each stage most of the datapath is unused.

5 Stages

- **I**nstr **F**etch
- **I**nstr **D**ecode
- **EX**ecute
- **MEM**ory Access
- **W**rite **B**ack Result

➪ Overlapped execution

```
lw $1,100($0)
lw $2,200($0)
```



How much better? (first, a detail …)

1.6 ns sequential,
1.0 ns overlapped.

© 2021 Dr. Muhammad Al-Hashimi
COPYRIGHT 1998-2005 MORGAN KAUFMANN PUBLISHERS, INC. ALL RIGHTS RESERVED

cs704fig_ilp1.cdr
Sunday, March 14, 2021 7:50:31 PM
Color profile: Disabled

Composite Default screen

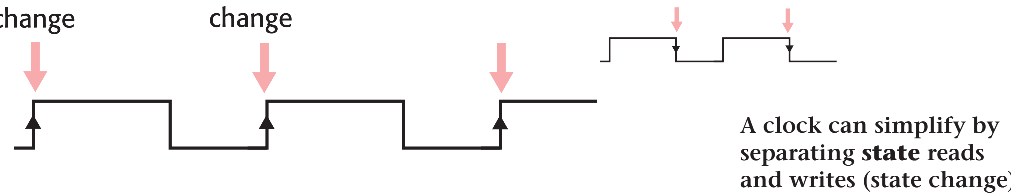
Review: Digital Logic Design
Clock Signal

A computer is a big **finite state machine** (FSM), a math tool to capture automatic logic.

A FSM transitions between different **states** in response to inputs (**Moore** machine assumed).

A physical FSM is built using **sequential logic** which directs reads of stored bits (encoding **state**), and writes in response to inputs, resulting in controlled moves between states.

- Sequential logic
- Edge triggered

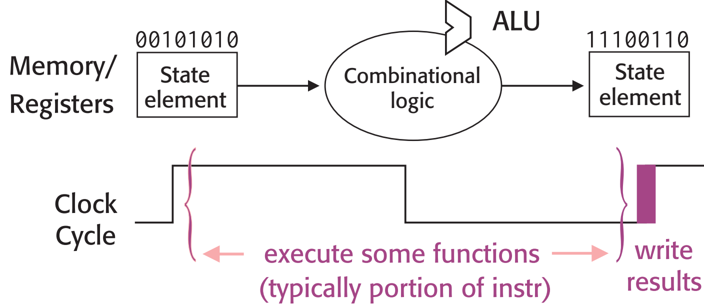


A clock can simplify by separating **state** reads and writes (state change).

Changes in input and output are said to be **synchronous**, i.e., on pre-agreed timing.

"State" elements

Combinational logic outputs a result depending only on inputs.



A fixed clock hides (masks) continuous, unpredictable physical timing of device/unit and path.

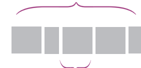
© 2021 Dr. Muhammad Al-Hashimi
COPYRIGHT 1998-2005 MORGAN KAUFMANN PUBLISHERS, INC. ALL RIGHTS RESERVED

cs704fig_ilp1.cdr
Sunday, March 14, 2021 7:50:31 PM
Color profile: Disabled

Composite Default screen

Processor Implementation
An Execution Pipeline

To fit lw, a sequential data-path can't have < 0.8 ns cycles.



Pipeline **clock cycle** set at 0.2 ns to accommodate slowest unit.

Quiz
What is the clock speed (frequency)?

Compare

✎ Time between instr

✎ Time to complete instr

✎ Speedup = ?

© 2021 Dr. Muhammad Al-Hashimi
COPYRIGHT 1998-2005 MORGAN KAUFMANN PUBLISHERS, INC. ALL RIGHTS RESERVED

cs704fig_ilp1.cdr
Sunday, March 14, 2021 7:50:31 PM
Color profile: Disabled

Composite Default screen

Execution Pipelines
Performance

- ➪ **Instr throughput**
- ➪ **Instr exec time**
- ➪ **Pipeline paradox**
- ➪ **Ideal pipeline**

Time between instr
(time to start or *issue* nth instr)

n	Time to issue non-pipelined	Time to issue pipelined	Speedup
4th	$.8 \times 3$	$.2 \times 3$	$\frac{.8 \times 3}{.2 \times 3} = 4$
11th	?	?	4

long run, complete instr
as fast as can issue them!

Time to complete
(total exec time)

For 1 instr there is no speedup!

n	Time to complete non-pipelined	Time to complete pipelined	Speedup
3	$.8 \times 3$	1.4	$\frac{2.4}{1.4} \approx 1.71$
10	$.8 \times 10$	2.8	$\frac{8.0}{2.8} \approx 2.86$
1003	$.8 \times 1003$	$.8 + 1003 \times .2$	$\frac{802.4}{201.4} \approx 3.98$
1,000,003	?	?	≈ 4.0

Quiz
What is the formula for the speedup in the bottom table? Answer next slide.

© 2021 Dr. Muhammad Al-Hashimi
COPYRIGHT 1998-2005 MORGAN KAUFMANN PUBLISHERS, INC. ALL RIGHTS RESERVED

cs704fig_ilp1.cdr
Sunday, March 14, 2021 7:50:31 PM
Color profile: Disabled

Composite Default screen

Pipeline Hazards

Actual speedup is less than ideal due to: 1) pipeline stages may not be perfectly balanced, and 2) one instr/cycle issue may not always be possible.

- **Prevent instr issue every cycle**
- **Types: depending on cause**
  - ✎ Structural: functional unit <u>conflict</u>
  - ✎ Data: operand <u>dependence</u>
  - ✎ Control: decision <u>dependance</u>

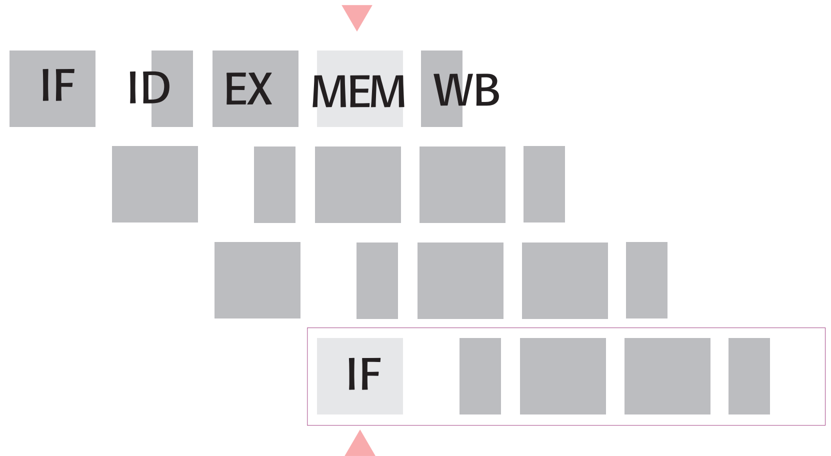
Speedup = time to compete instr (non-pipelined) / time to complete instr (pipelined). Note in pipelined case, 1st instr goes through m-1 stages after which an instr will complete per cycle, i.e., n instr complete in 0.8 + n×0.2, therefore 10 instr complete after 0.8 + 10×0.2 = 2.8 ns in 4+n = 4+10 =14 cycles.

© 2021 Dr. Muhammad Al-Hashimi
COPYRIGHT 1998-2005 MORGAN KAUFMANN PUBLISHERS, INC. ALL RIGHTS RESERVED

cs704fig_ilp1.cdr
Sunday, March 14, 2021 7:50:31 PM
Color profile: Disabled

Composite Default screen

Structural Hazards



Separate physical memories remove the conflict.

➪ **Datapath structure causes the hazard**

Typically a 1-time cost at design time.

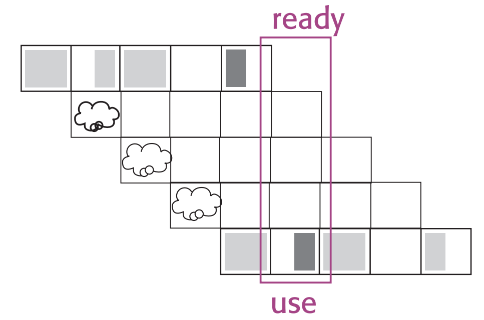
➪ **Resolving structural hazards**

© 2021 Dr. Muhammad Al-Hashimi
COPYRIGHT 1998-2005 MORGAN KAUFMANN PUBLISHERS, INC. ALL RIGHTS RESERVED

Data Hazards

➪ Stall cycle (bubble)

➪ Example

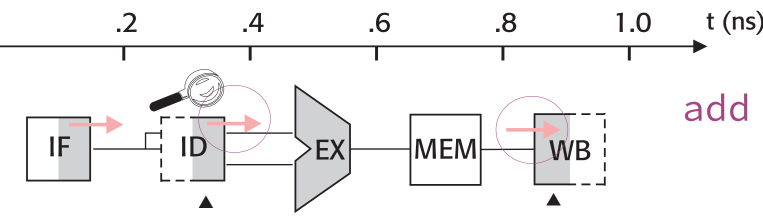


The data hazard seems to cause 3 delay cycles more than a **hazard-free** sequence, each a missed opportunity for more parallelism.

```
add $s0,$t0,$t1
sub $t2,$s0,$t3
```

➪ Pipeline representation

Quiz
The **read-after-write** data hazard will actually cause 2c delay in MIPS. Why?



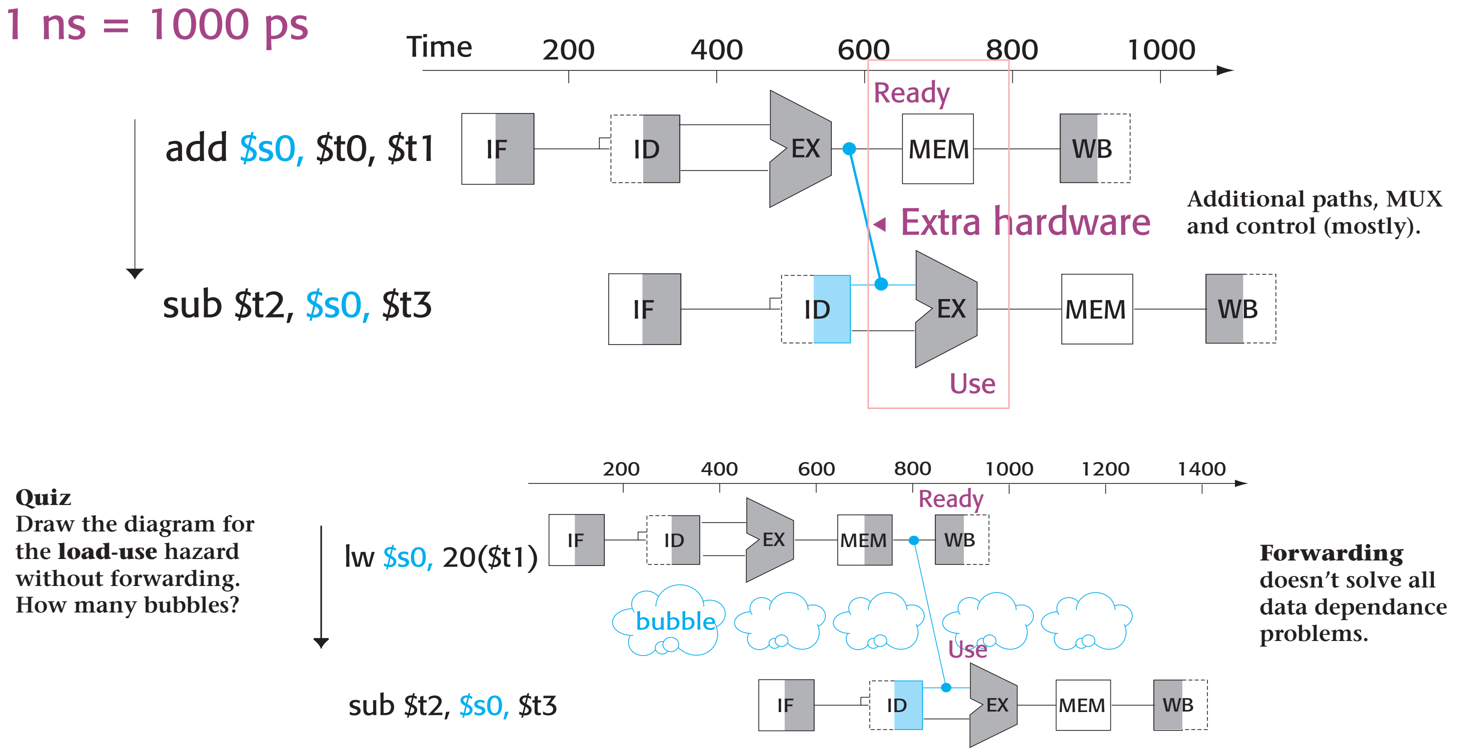
© 2021 Dr. Muhammad Al-Hashimi
COPYRIGHT 1998-2005 MORGAN KAUFMANN PUBLISHERS, INC. ALL RIGHTS RESERVED

cs704fig_ilp1.cdr
Sunday, March 14, 2021 7:50:31 PM
Color profile: Disabled

Composite Default screen

Resolving Pipeline Hazards
Forward Results

➪ Bypssing



Additional paths, MUX and control (mostly).

Quiz
Draw the diagram for the **load-use** hazard without forwarding. How many bubbles?

Forwarding doesn't solve all data dependance problems.

© 2021 Dr. Muhammad Al-Hashimi
COPYRIGHT 1998-2005 MORGAN KAUFMANN PUBLISHERS, INC. ALL RIGHTS RESERVED

cs704fig_ilp1.cdr
Sunday, March 14, 2021 7:50:31 PM
Color profile: Disabled

Composite Default screen

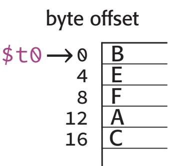
Resolving Pipeline Hazards
Reorder Instructions

➪ Cycles-per-instruction (CPI)

High-level code compiled into a "sensible" (symbolic) machine code.

```
$t3   $t1    $t2
   A = B + E;
   C = B + F;
$t5        $t4
```

Assume variables stored sequentially in memory starting at address loaded in `$t0`




```
lw   $t1, 0($t0)     # fetch B
lw   $t2, 4($t0)     # fetch E
add  $t3, $t1,$t2    # A=B+E
sw   $t3, 12($t0)    # store A
lw   $t4, 8($t0)     # fetch F
add  $t5, $t1,$t4    # C=B+F
sw   $t5, 16($t0)    # store C
```

Quiz
What's the CPI in a perfect pipeline in the long run?

Exercise
1. Draw a pipeline diagram, determine number of cycles to complete sequence in each case
   (a) Without forwarding or reordering (worst case). Hint: 5 data hazards.
   (b) With forwarding alone (hardware-only solution)
   (c) With both forwarding and reordering (pipeline-aware compiler)
2. What's the **CPI** for ideal **hazard-free** pipeline?
3. What is the **speedup** in each case? (CPI case/CPI hazard-free)

© 2021 Dr. Muhammad Al-Hashimi
COPYRIGHT 1998-2005 MORGAN KAUFMANN PUBLISHERS, INC. ALL RIGHTS RESERVED

cs704fig_ilp1.cdr
Sunday, March 14, 2021 7:50:32 PM
Color profile: Disabled

Composite Default screen

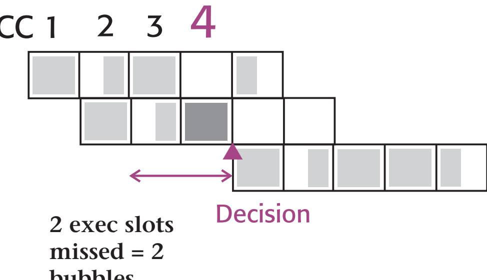
Control Hazards

➪ **Delay slot**

Stall on branch

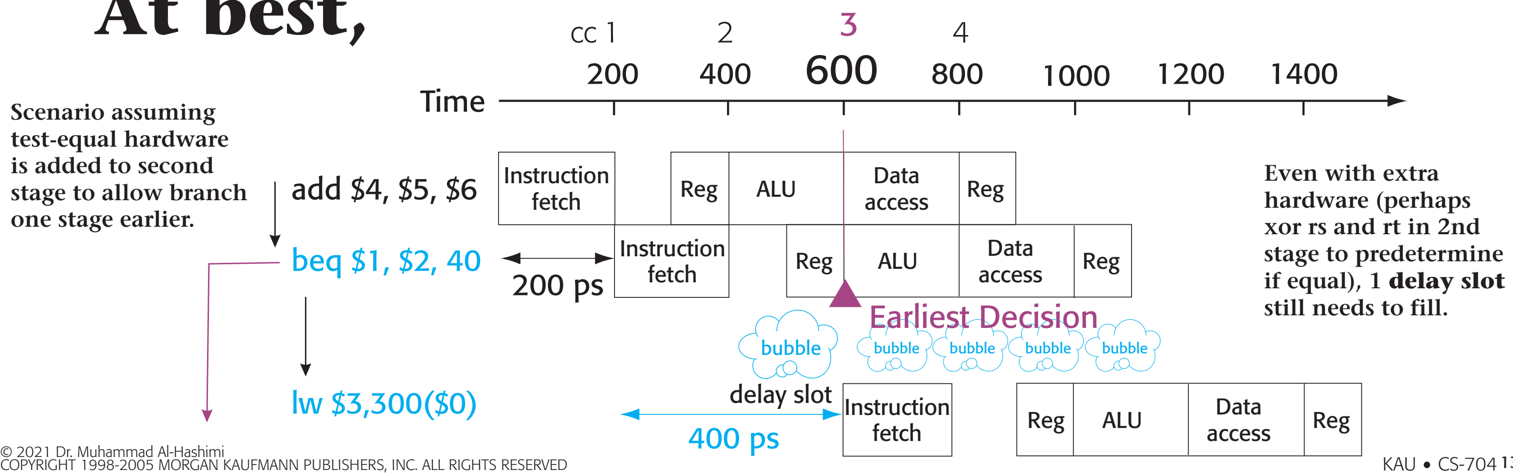
Test difference $1−$2 for zero (ALU op) so must wait until clock cycle 4 to know if branch taken.

```
add $4,$5,$6
▶ beq $1,$2,40
lw $3,300($0)
```



2 exec slots missed = 2 bubbles

At best,

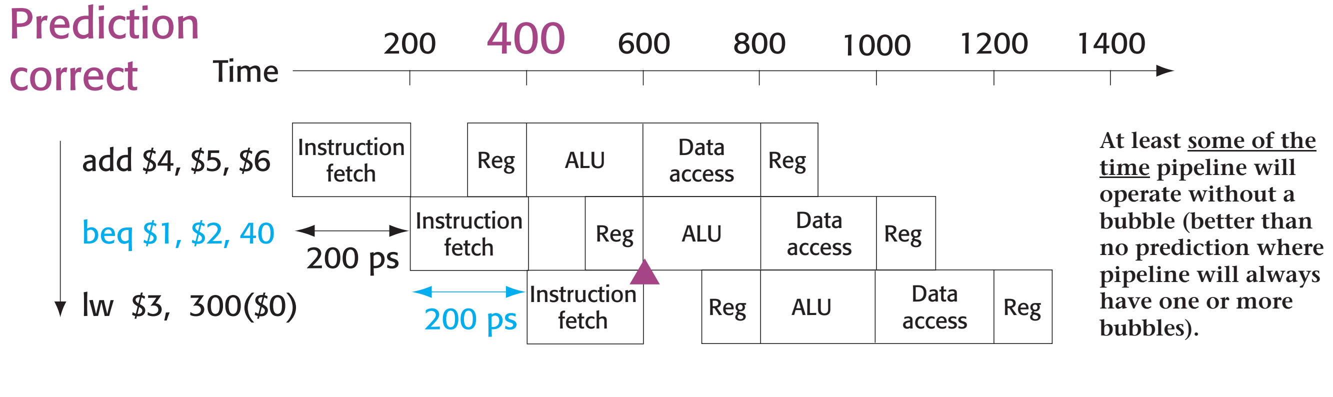


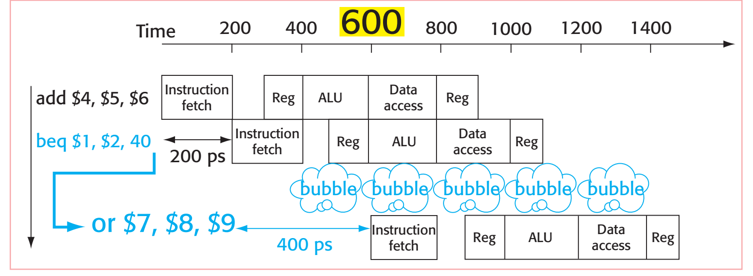
© 2021 Dr. Muhammad Al-Hashimi
COPYRIGHT 1998-2005 MORGAN KAUFMANN PUBLISHERS, INC. ALL RIGHTS RESERVED

cs704fig_ilp1.cdr
Sunday, March 14, 2021 7:50:32 PM
Color profile: Disabled
Composite Default screen

Resolving Pipeline Hazards
Branch Prediction

Predict: branch always untaken





© 2021 Dr. Muhammad Al-Hashimi
COPYRIGHT 1998-2005 MORGAN KAUFMANN PUBLISHERS, INC. ALL RIGHTS RESERVED

cs704fig_ilp1.cdr
Sunday, March 14, 2021 7:50:32 PM
Color profile: Disabled

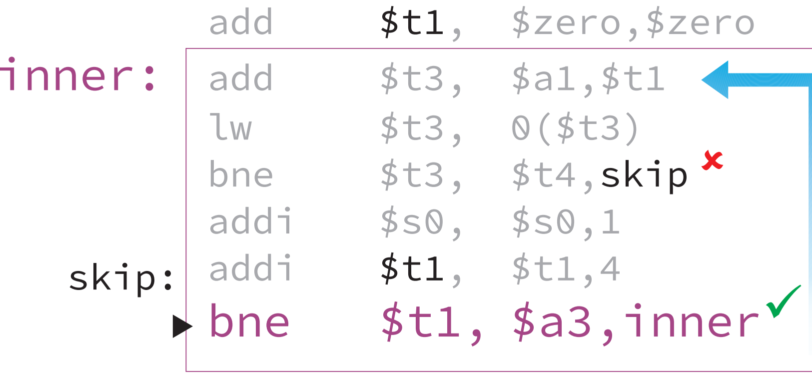
Composite Default screen

Better Prediction

➪ Basic block

Common looping pattern

Example **Basic block** depicting a typical loop. Refer to Glossary for definition.



Basic blocks offer opportunities to overcome pipeline hazards.

Typical loop, n taken : 1 not taken

Predict: back branch always taken

© 2021 Dr. Muhammad Al-Hashimi
COPYRIGHT 1998-2005 MORGAN KAUFMANN PUBLISHERS, INC. ALL RIGHTS RESERVED

cs704fig_ilp1.cdr
Sunday, March 14, 2021 7:50:32 PM
Color profile: Disabled

Composite Default screen

Resolving Control Hazards

Simple **prediction** (guessing) improves pipeline performance by removing bubbles in some cases, better guessing should improve even more!

- **Branch prediction**
  - Fixed (static) prediction
  - Dynamic (changing) prediction

Reordering instructions to avoid data and control hazards is a form of compiler-assissted resolution.

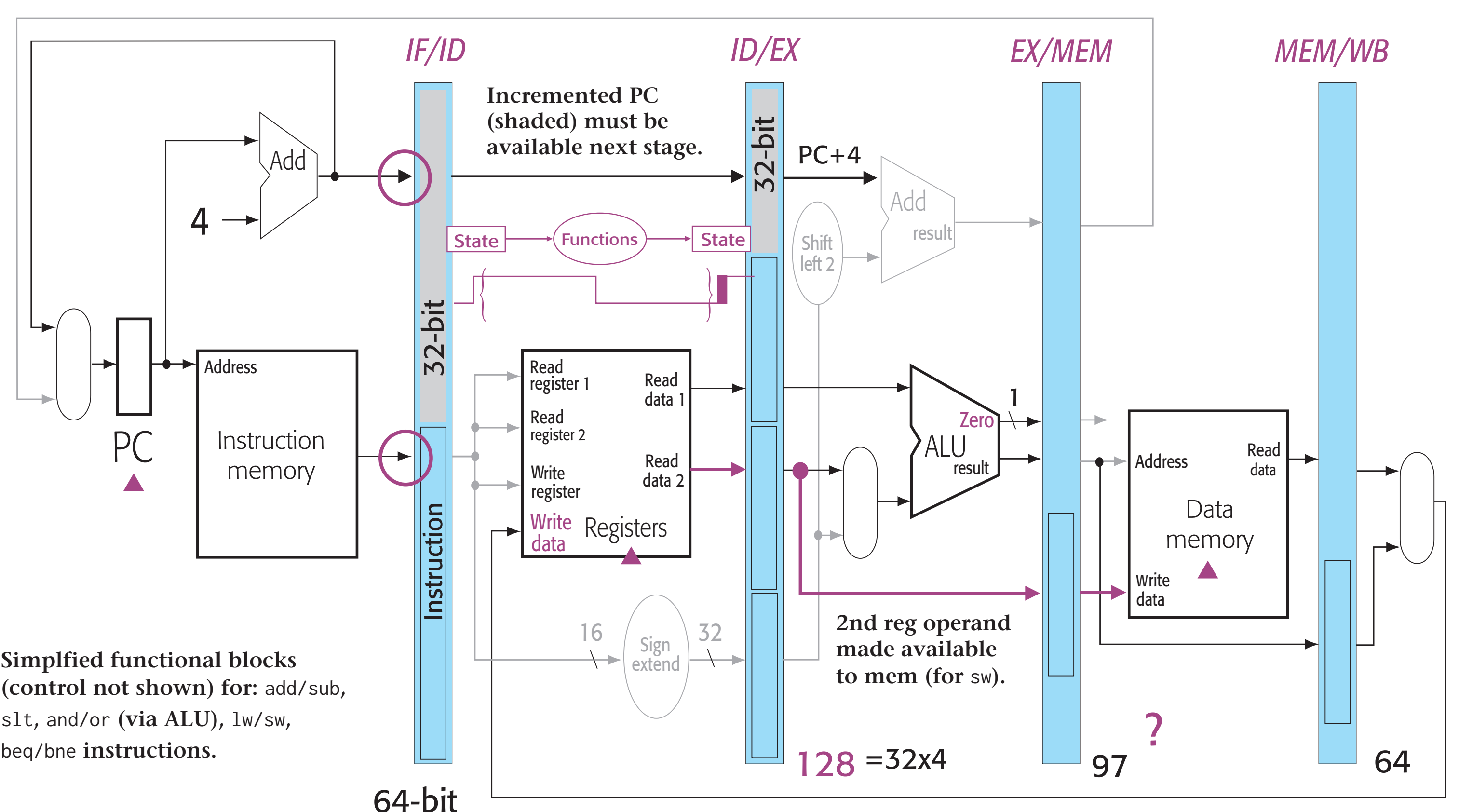
- **Compiler-assisted resolution**
- **Speculative execution (later)**

© 2021 Dr. Muhammad Al-Hashimi
COPYRIGHT 1998-2005 MORGAN KAUFMANN PUBLISHERS, INC. ALL RIGHTS RESERVED

cs704fig_ilp1.cdr
Sunday, March 14, 2021 7:50:32 PM
Color profile: Disabled

Composite Default screen

A Simple MIPS Datapath



Simplfied functional blocks (control not shown) for: `add/sub`, `slt`, `and/or` **(via ALU),** `lw/sw`, `beq/bne` **instructions.**

© 2021 Dr. Muhammad Al-Hashimi
COPYRIGHT 1998-2005 MORGAN KAUFMANN PUBLISHERS, INC. ALL RIGHTS RESERVED

cs704fig_ilp1.cdr
Sunday, March 14, 2021 7:50:32 PM
Color profile: Disabled

Composite Default screen

How it Works

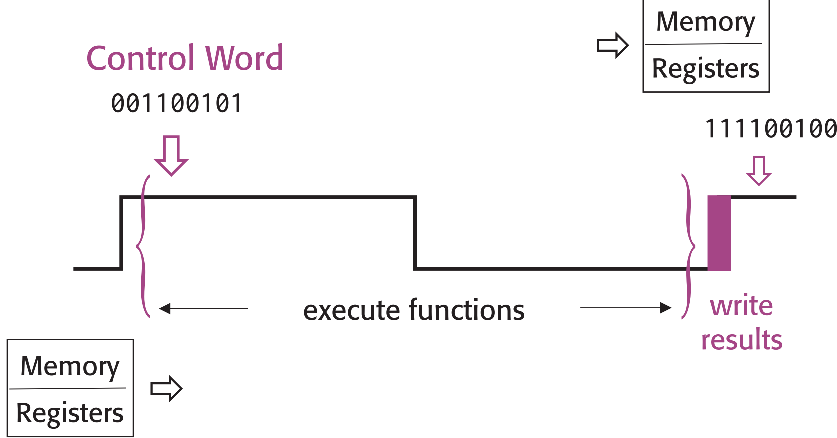
⇨ Program state

Electronic signals will propagate in all circuits and invalid bits may occur in datapath.

Control unit outputs a suitable **control word** in each cycle in response to opcode.

Sequence of control words needed to exec instruction may be generated on-the-fly by fixed logic, or stored in a memory (**microcode**).

add ▸ 001100101
111100100
...
lw ▸ 001100101
...



A control word ensures desired functions will compute correctly, and only desired results are recorded at end of each cycle either in stage or **programmer-visible** registers, or in memory.

© 2021 Dr. Muhammad Al-Hashimi
COPYRIGHT 1998-2005 MORGAN KAUFMANN PUBLISHERS, INC. ALL RIGHTS RESERVED

cs704fig_ilp1.cdr
Sunday, March 14, 2021 7:50:32 PM
Color profile: Disabled

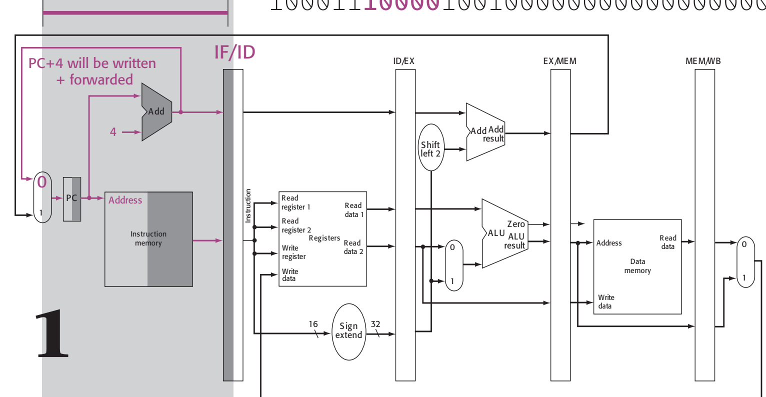
Composite Default screen

`lw $17,0($16)`

Instruction fetch

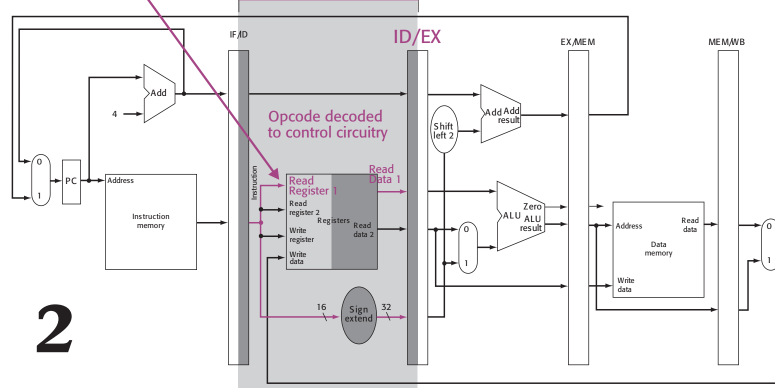
Instr[26-21]

100011**10000**100100000000000000000

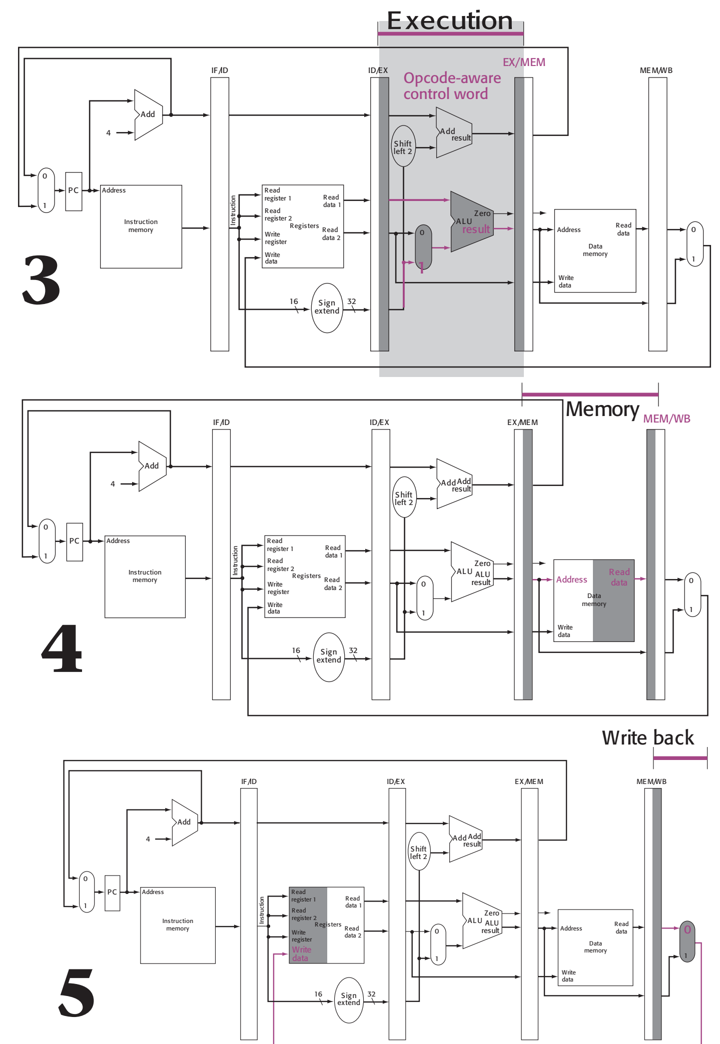


Instr[26-21]

Instruction decode



Execution



© 2021 Dr. Muhammad Al-Hashimi
COPYRIGHT 1998-2005 MORGAN KAUFMANN PUBLISHERS, INC. ALL RIGHTS RESERVED

cs704fig_ilp1.cdr
Sunday, March 14, 2021 7:50:32 PM
Color profile: Disabled

Composite Default screen

Essay Assignment

Check *Essay Assignments* topic in the course group for essay descriptions.

Check datapath case study links in the Reading File (under Background).

Pick one

- ✎ Review ARM pipelines, compare-contast with classic 5-stage MIPS
- ✎ Discuss how well classic MIPS fits RISC criteria

Expectations

Check the group for specifications.

Contribute small essay in course group

© 2021 Dr. Muhammad Al-Hashimi
COPYRIGHT 1998-2005 MORGAN KAUFMANN PUBLISHERS, INC. ALL RIGHTS RESERVED

cs704fig_ilp1.cdr
Sunday, March 14, 2021 7:50:32 PM
Color profile: Disabled

Composite Default screen

Processor Instructions Design to Pipeline

“ Instr sets can either simplify or make life harder for pipeline designers ” P&H

Exercise
Explain the impact of each feature on pipeline, give examples from MIPS.

- **Fixed length**
- **Regular (few, similar formats)**

  Operands in the same places make **pre-fetch** possible, that's, decode independent of opcode.
- **Load-store architecture**
- **Memory operand alignment**
- **Single final write back**

© 2021 Dr. Muhammad Al-Hashimi
COPYRIGHT 1998-2005 MORGAN KAUFMANN PUBLISHERS, INC. ALL RIGHTS RESERVED

cs704fig_ilp1.cdr
Sunday, March 14, 2021 7:50:32 PM
Color profile: Disabled
Composite Default screen

Concept Review

Pipeline Performance

Performance ultimately depends on how well **conflicts** (=hazards) are handled.

- **Dependencies in instruction stream execution cause delay**
- **Conflict types**
  - Datapath resource
  - Program dataflow
  - Program decision

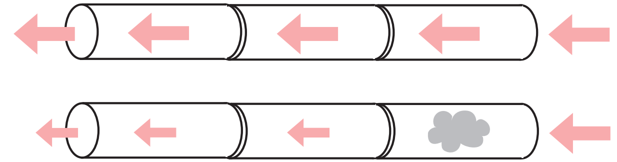
© 2021 Dr. Muhammad Al-Hashimi
COPYRIGHT 1998-2005 MORGAN KAUFMANN PUBLISHERS, INC. ALL RIGHTS RESERVED

cs704fig_ilp1.cdr
Sunday, March 14, 2021 7:50:32 PM
Color profile: Disabled

Composite Default screen

Concept Review
Pipelined Execution

A "pipeline" is created by overlapping instr exec.



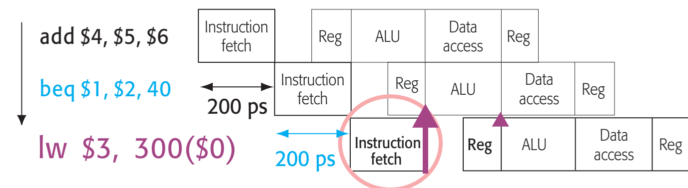
Flow is even in a perfect execution pipeline, like a real pipeline, each pipe contributes equally to the flow.

Every cycle a new instruction isn't started is a missed opportunity. Stall cycles, like bubbles in a real pipeline, reduce the **throughput** of an execution pipeline.

➪ Instr latency

➪ Misprediction cost

Cycles spent on wrong instructions are wasted, pipeline must be *flushed*, i.e., wasted cycles.

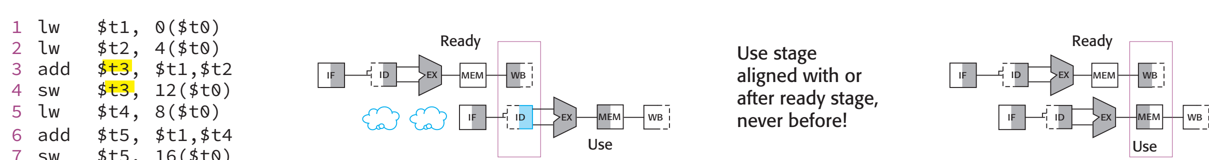


© 2021 Dr. Muhammad Al-Hashimi
COPYRIGHT 1998-2005 MORGAN KAUFMANN PUBLISHERS, INC. ALL RIGHTS RESERVED

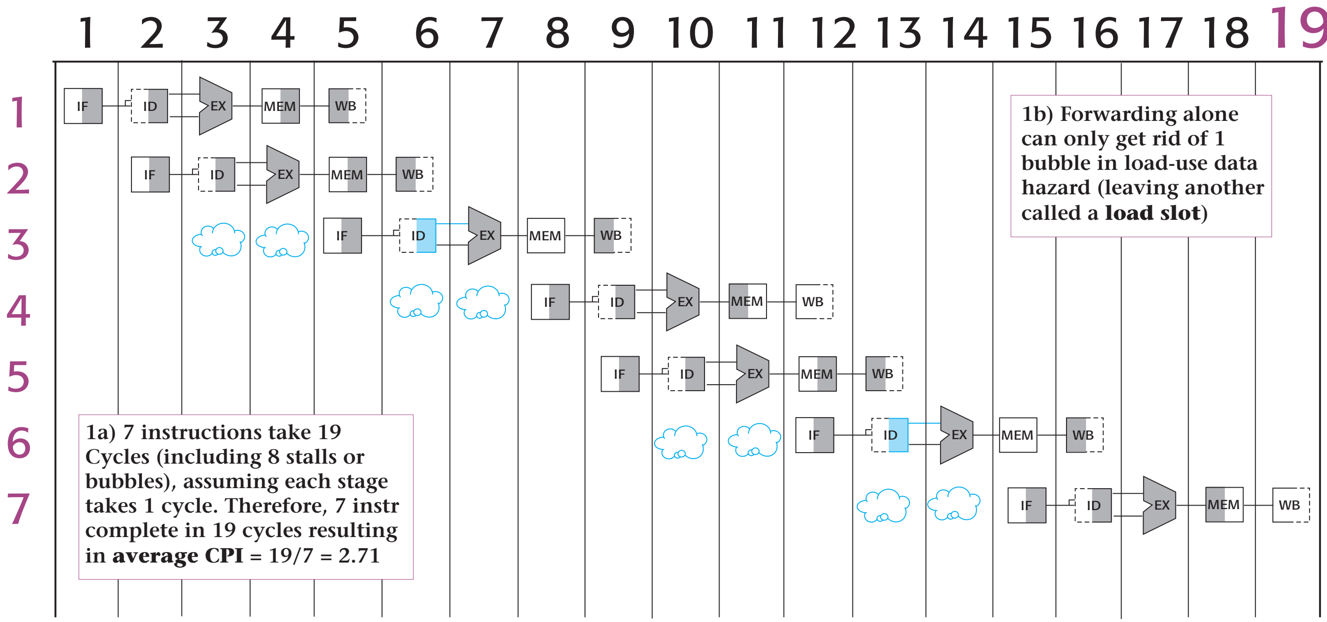
cs704fig_ilp1.cdr
Sunday, March 14, 2021 7:50:33 PM
Color profile: Disabled

Composite Default screen

Resolving Hazards Exercise



Original note write-after-write hazard



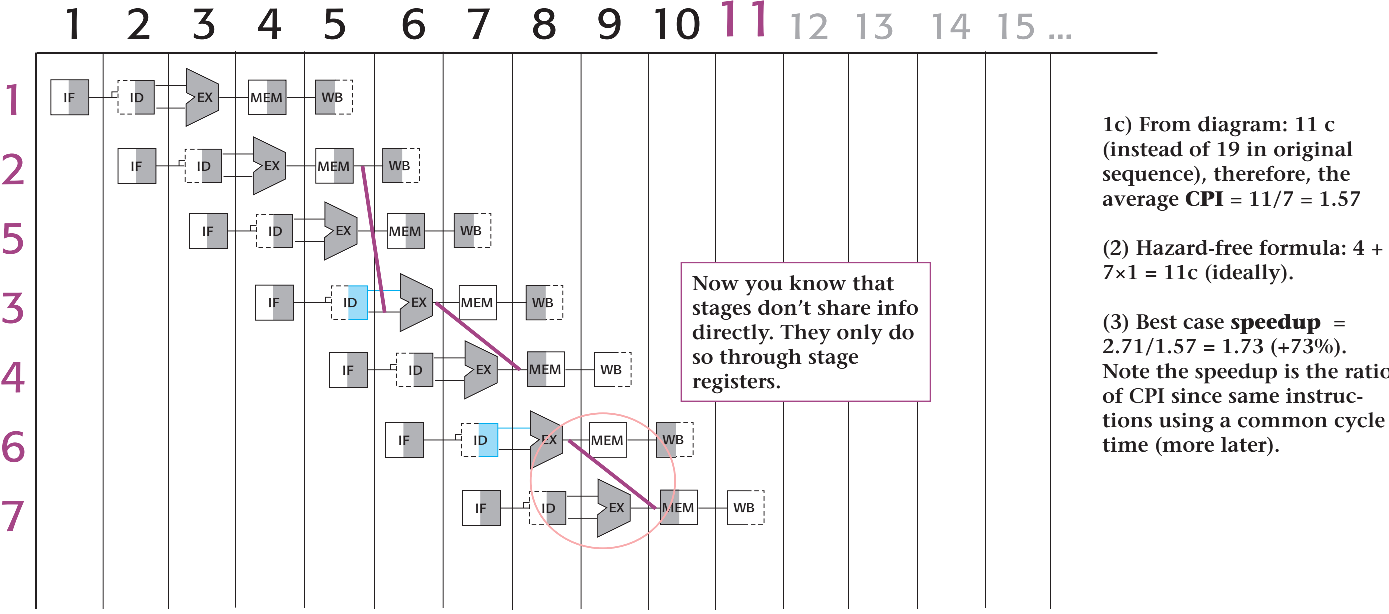
© 2021 Dr. Muhammad Al-Hashimi
COPYRIGHT 1998-2005 MORGAN KAUFMANN PUBLISHERS, INC. ALL RIGHTS RESERVED

Resolving Hazards Exercise

```
1 lw   $t1, 0($t0)
2 lw   $t2, 4($t0)
5 lw   $t4, 8($t0)
3 add  $t3, $t1,$t2
4 sw   $t3, 12($t0)
6 add  $t5, $t1,$t4
7 sw   $t5, 16($t0)
```

Re-ordered with forwarding

Execution timings and CPI here are ideal. We will see why CPI must be measured carefully when we factor in memory.



1c) From diagram: 11 c (instead of 19 in original sequence), therefore, the average **CPI** = 11/7 = 1.57

(2) Hazard-free formula: 4 + 7×1 = 11c (ideally).

(3) Best case **speedup** = 2.71/1.57 = 1.73 (+73%). Note the speedup is the ratio of CPI since same instructions using a common cycle time (more later).

© 2021 Dr. Muhammad Al-Hashimi
COPYRIGHT 1998-2005 MORGAN KAUFMANN PUBLISHERS, INC. ALL RIGHTS RESERVED