

William Stallings

Computer Organization and Architecture

Chapter 11

CPU Structure and Function

CPU Structure

⌘ CPU must:

- ☐ Fetch instructions
- ☐ Interpret instructions
- ☐ Fetch data
- ☐ Process data
- ☐ Write data

Registers

- ⌘ CPU must have some working space (temporary storage)
- ⌘ Called registers
- ⌘ Number and function vary between processor designs
- ⌘ One of the major design decisions
- ⌘ Top level of memory hierarchy

User Visible Registers

⌘ General Purpose

⌘ Data

⌘ Address

⌘ Condition Codes

General Purpose Registers (1)

- ⌘ May be true general purpose
- ⌘ May be restricted
- ⌘ May be used for data or addressing
- ⌘ Data
 - ☒ Accumulator
- ⌘ Addressing
 - ☒ Segment

General Purpose Registers (2)

⌘ Make them general purpose

- ☑ Increase flexibility and programmer options
- ☑ Increase instruction size & complexity

⌘ Make them specialized

- ☑ Smaller (faster) instructions
- ☑ Less flexibility

How Many GP Registers?

- ⌘ Between 8 - 32
- ⌘ Fewer = more memory references
- ⌘ More does not reduce memory references and takes up processor real estate
- ⌘ See also RISC

How big?

- ⌘ Large enough to hold full address
- ⌘ Large enough to hold full word
- ⌘ Often possible to combine two data registers
 - ☑ C programming
 - ☑ `double int a;`
 - ☑ `long int a;`

Condition Code Registers

- ⌘ Sets of individual bits

 - ☐ e.g. result of last operation was zero

- ⌘ Can be read (implicitly) by programs

 - ☐ e.g. Jump if zero

- ⌘ Can not (usually) be set by programs

Control & Status Registers

- ⌘ Program Counter
 - ⌘ Instruction Decoding Register
 - ⌘ Memory Address Register
 - ⌘ Memory Buffer Register
-
- ⌘ Revision: what do these all do?

Program Status Word

- ⌘ A set of bits
- ⌘ Includes Condition Codes
- ⌘ Sign of last result
- ⌘ Zero
- ⌘ Carry
- ⌘ Equal
- ⌘ Overflow
- ⌘ Interrupt enable/disable
- ⌘ Supervisor

Supervisor Mode

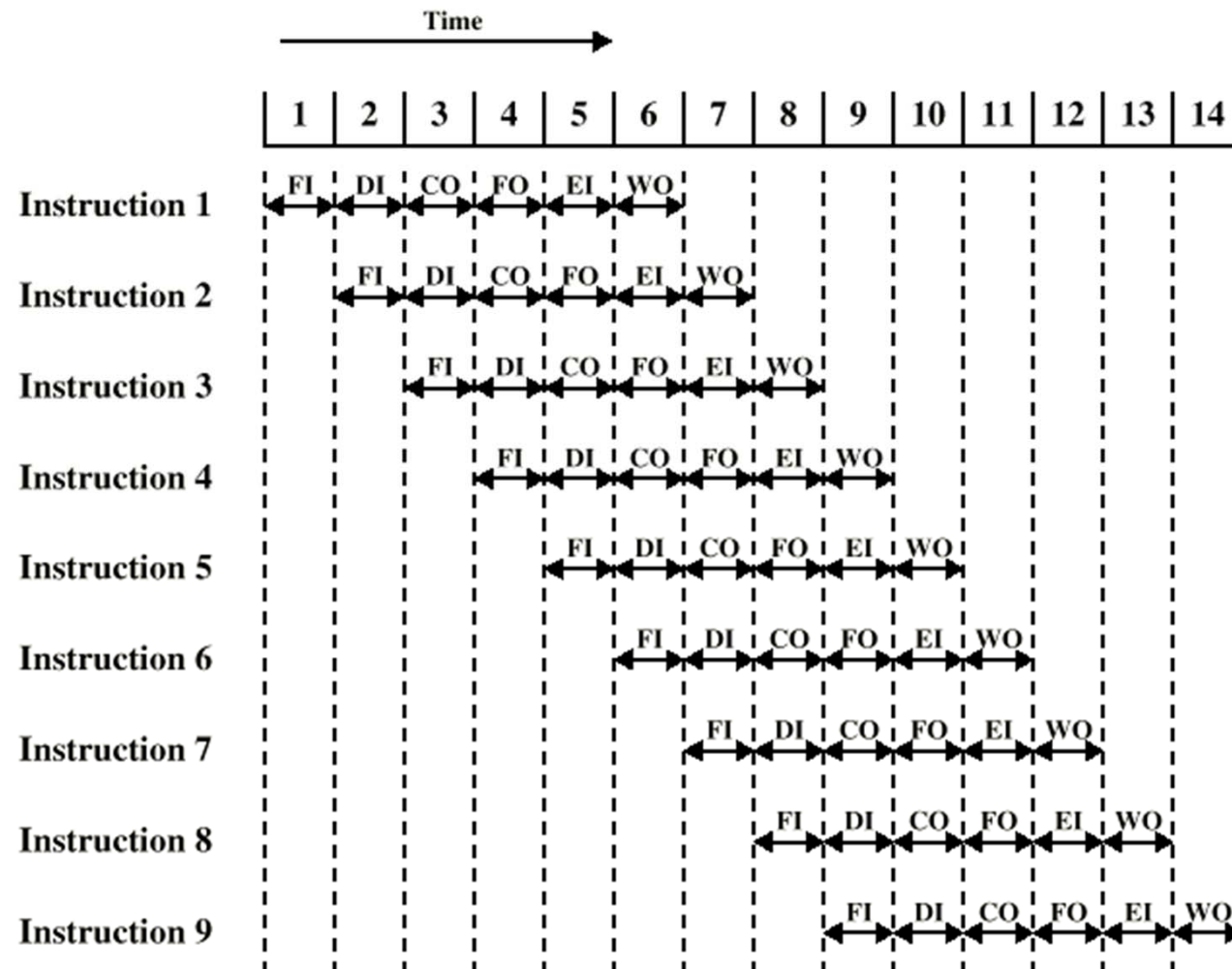
- ⌘ Intel ring zero
- ⌘ Kernel mode
- ⌘ Allows privileged instructions to execute
- ⌘ Used by operating system
- ⌘ Not available to user programs

Pipelining

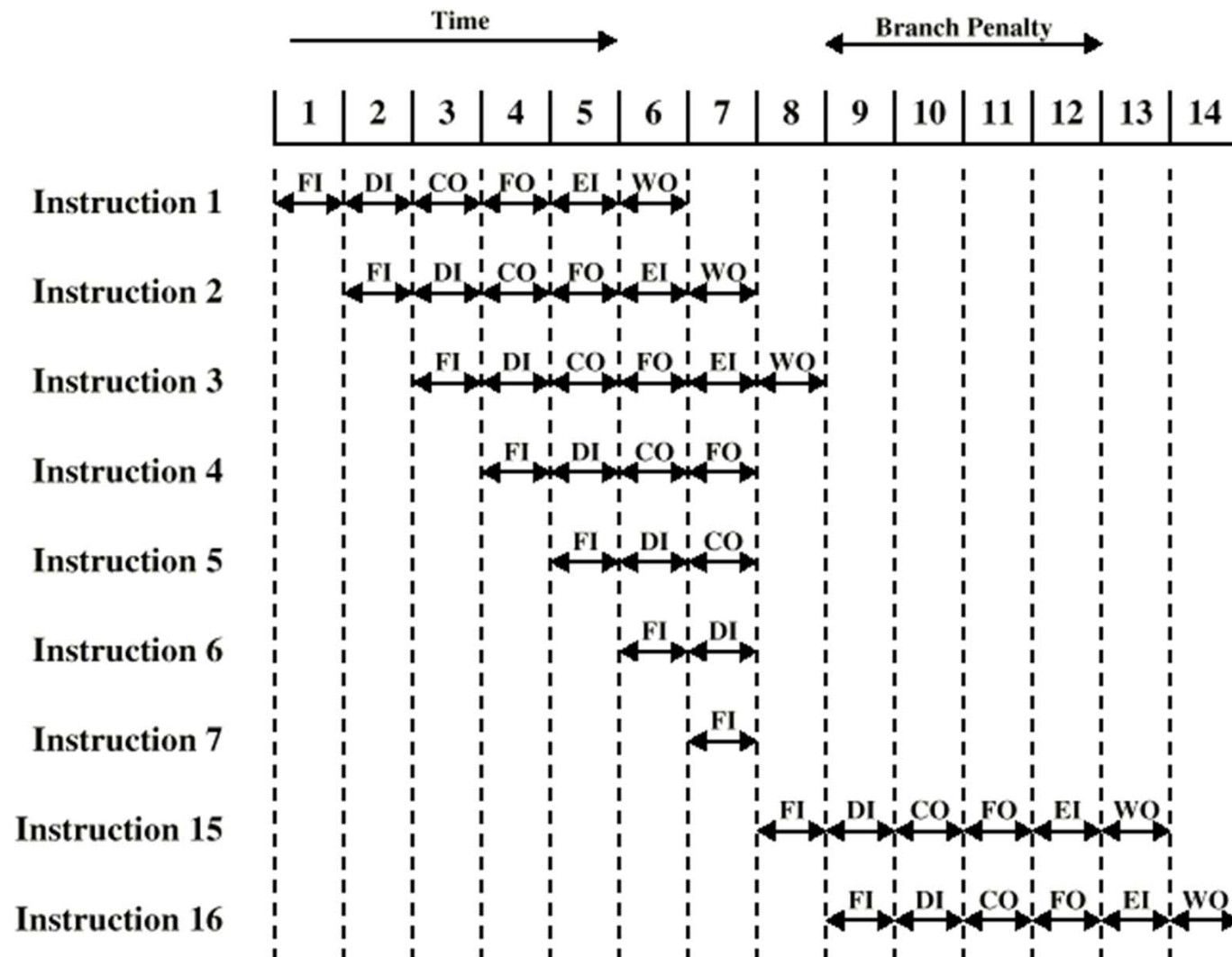
- ⌘ Fetch instruction
- ⌘ Decode instruction
- ⌘ Calculate operands (i.e. EAs)
- ⌘ Fetch operands
- ⌘ Execute instructions
- ⌘ Write result

- ⌘ Overlap these operations

Timing of Pipeline



Branch in a Pipeline



Dealing with Branches

- ⌘ Multiple Streams (Fluxos)
- ⌘ Prefetch Branch Target
- ⌘ Loop buffer
- ⌘ Branch prediction
- ⌘ Delayed branching

Multiple Streams

- ⌘ Have two pipelines
 - ⌘ Prefetch each branch into a separate pipeline
 - ⌘ Use appropriate pipeline
-
- ⌘ Leads to bus & register contention
 - ⌘ Multiple branches lead to further pipelines being needed

Prefetch Branch Target

- ⌘ Target of branch is prefetched in addition to instructions following branch
- ⌘ Keep target until branch is executed
- ⌘ Used by IBM 360/91

Loop Buffer

- ⌘ Very fast memory
- ⌘ Maintained by fetch stage of pipeline
- ⌘ Check buffer before fetching from memory
- ⌘ Very good for small loops or jumps
- ⌘ c.f. cache
- ⌘ Used by CRAY-1

Branch Prediction (1)

⌘ Predict never taken

- ☒ Assume that jump will not happen
- ☒ Always fetch next instruction
- ☒ 68020 & VAX 11/780
- ☒ VAX will not prefetch after branch if a page fault would result (O/S v CPU design)

⌘ Predict always taken

- ☒ Assume that jump will happen
- ☒ Always fetch target instruction

Branch Prediction (2)

⌘ Predict by Opcode

- ☑ Some instructions are more likely to result in a jump than others
- ☑ Can get up to 75% success

⌘ Taken/Not taken switch

- ☑ Based on previous history
- ☑ Good for loops

Branch Prediction (3)

⌘ Delayed Branch

- ☑ Do not take jump until you have to
- ☑ Rearrange instructions