Machine Instructions

Instruction Set Architectures

Definition

An **instruction set architecture**, or "ISA", defines the hardware components and behaviors visible to the programmer.

- An ISA provides all information required to write a program in **machine language**.
  - Memory access and limits
  - Registers and their purposes
  - Supported data types
  - The **instruction set**
- ISAs allow machine language programmers to write code for different hardware, as long as it conforms to the same ISA.

Definition

A **microarchitecture** details a hardware implementation of an ISA.

Machine Instructions

Definition

The **instruction** is the computer's fundamental unit of work.

- The instruction's **opcode** specifies which operation to perform.
- The instruction's **operands** specify data for the operation.

- An instruction is encoded as a sequence of bits. (Just like data!)
  - Interpreted by the Control Unit.
  - Ex: 0001010001000001 - This instructs the LC-3 to add two integer values together.

Machine Instructions

- The LC-3 uses 16-bit instructions with 4-bit opcodes. How many possible opcodes?
- The LC-3 has 8 registers. How many bits are needed to specify a register operand?

Example

- Format of an `ADD`:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
`0001 (ADD)`				`DestR`			`Src1`			0	0	0	`Src2`		

- Instance of an `ADD`, representing "`R6 = R2 + R5`":

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	1	0	0	1	0	0	0	0	1	0	1

Instruction Cycles

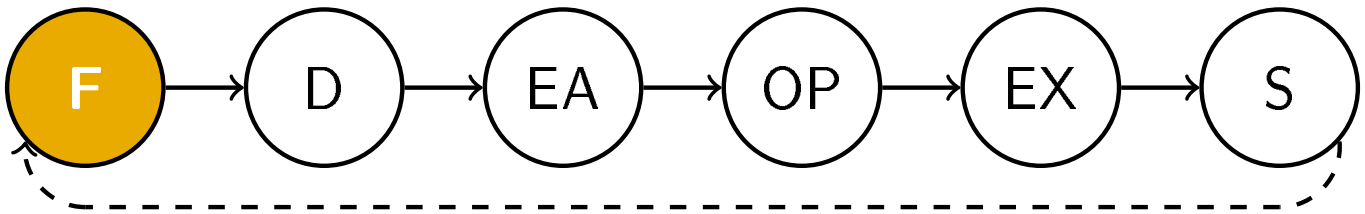
Definition

The **instruction cycle** is a series of sequential **stages** in which a computer executes a single machine instruction.

- Each CPU manufacturer has their own unique instruction cycles.
- The LC-3 processes instructions in 6 stages:

1. Fetch
2. Decode
3. Evaluate Address
4. Fetch Operands
5. Execute
6. Store

Fetch

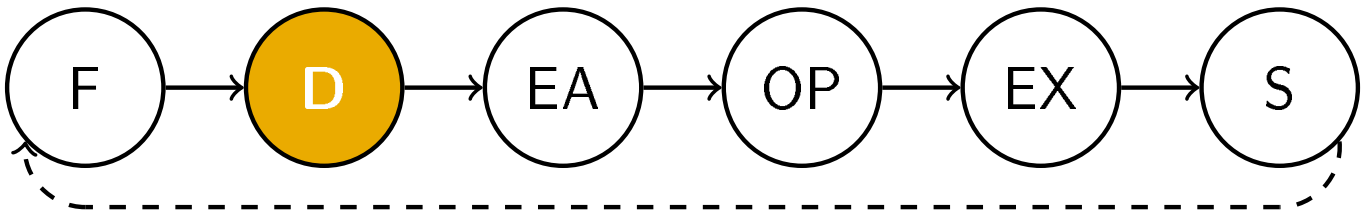


- The Fetch stage loads the instruction from memory.

1. Copy the contents of the PC into the MAR.
2. Send a 'read' signal to memory.
3. Copy the contents of the MDR into the IR.
4. Increment the PC (PC = PC + 1).

- The Fetch stage is identical for all instructions.

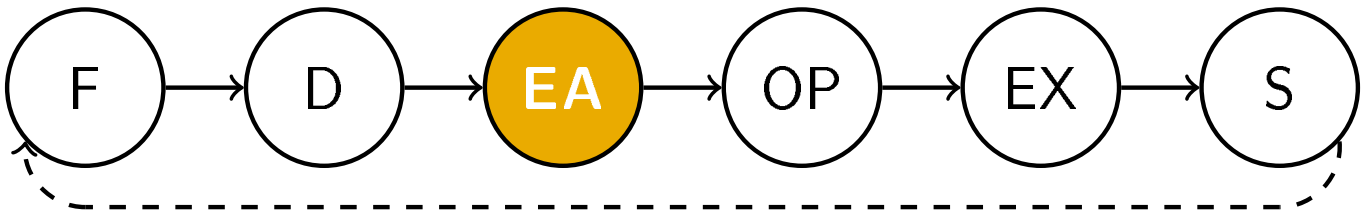
Decode



- The Decode stage determines how to execute the instruction.

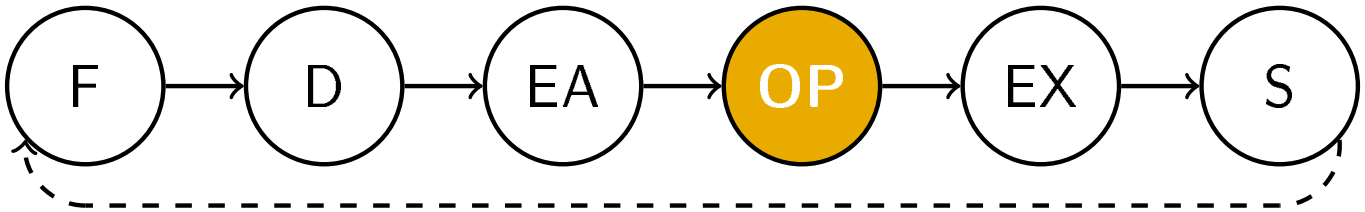
1. Identify the opcode.
2. Assert control line based on 4-to-16 decoder.
3. Identify the operands based on the opcode.
   - The operands vary by instruction.

Evaluate Address



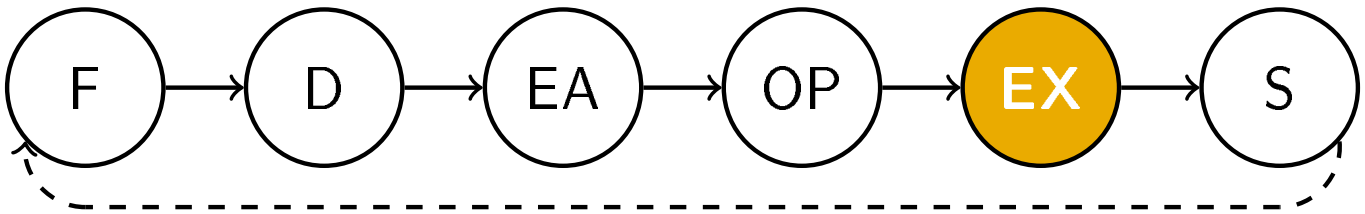
- The Evaluate Address stage computes an address used to access memory.
- The hardware operations performed during the Evaluate Address stage vary by instruction.
- Not every instruction uses the Evaluate Address stage.
  - For example, the add instruction from the previous slide does not access memory at all.

Fetch Operands



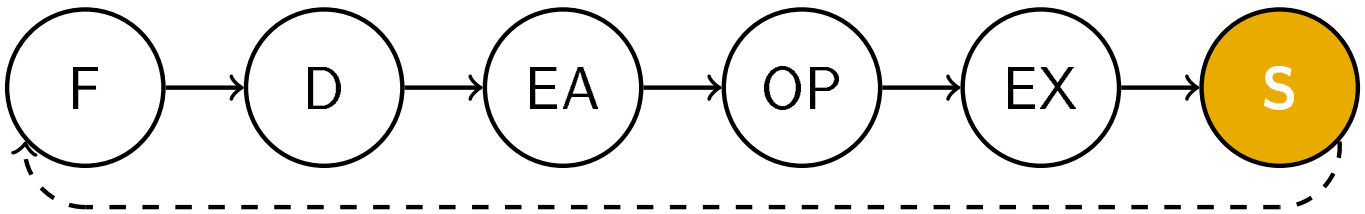
- The Fetch Operands stage loads the instruction's operands.
- The operations performed during the Fetch Operands stage vary by instruction.
- Some instructions load values from memory during this stage; others load values from registers.

Execute



- The Execute stage performs operations that require the ALU.
- Send a signal to the ALU (either “add”, “and”, or “not”).
- Not every instruction uses the Execute stage.

Store



- The Store stage writes the result of the instruction to its destination.
- The hardware operations performed during the Store stage vary by instruction.
- Some instructions store values in memory during this stage; others store values in registers.

LC-3 Instructions

- Broadly speaking, the LC-3 has three types of instructions.

Computational	Data Movement	Control
ADD	LD	BR
AND	LDI	JMP/RET
NOT	LDR	JSR/JSRR
	LEA	RTI
	ST	TRAP
	STI	
	STR	

- Computational instructions compute values.
- Data movement instructions move data into or out of memory.
- Control instructions modify the PC.

Computational Instructions

Example

- Consider an `ADD` instruction:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0001 (ADD)				Dest			Src1			0	0	0	Src2		

1. (F) Load the instruction from memory.
2. (D) Identify `Dest`, `Src1`, and `Src2`.
3. (EA) *n/a*
4. (OP) Retrieve values from `Src1` and `Src2` registers.
5. (EX) Add the values of `Src1` and `Src2`.
6. (S) Place the sum into `Dest`.

Data Movement Instructions

Example

- Consider an `LDR` instruction:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0110 (LDR)				DestR			BaseR			Offset					

1. (F) Load the instruction from memory.
2. (D) Identify `DestR` and `BaseR`; extract `Offset`.
3. (EA) Add `Offset` to the value of `BaseR`.
4. (OP) Load the value at the resulting address from memory.
5. (EX) *n/a*
6. (S) Place the loaded value into `DestR`.

Control Instructions

- In the FETCH stage, the PC is incremented by 1. What does this mean about the next instruction that will be run?
- When, in a high level language (HLL), do we not necessarily want to run the next line of code?
- Control instructions modify the PC.
  - Jump instructions are unconditional. HLL examples?
  - Branch instructions are conditional. HLL examples?

Control Instructions

Example

- Consider a `JMP` instruction:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1100 (JMP)				0	0	0	BaseR			0	0	0	0	0	0

1. (F) Load the instruction from memory.
2. (D) Identify `BaseR`.
3. (EA) Retrieve the value of `BaseR`.
4. (OP) *n/a*
5. (EX) *n/a*
6. (S) Place the value of `BaseR` into the PC.

Instruction Pipelines

Definition

Pipelining overlaps instructions, executing multiple instructions at different stages simultaneously.

- Not every instruction needs every stage.
- Each stage may take multiple clock cycles. What is a clock cycle?

Example

- ARM processors have an 8-stage pipeline.
- Apple A11 processors have a 16-stage pipeline.
- Most Intel Core processors have a 14-stage pipeline.
- The LC-3 **does not** implement pipelining.