Computer Organization (IS F242)

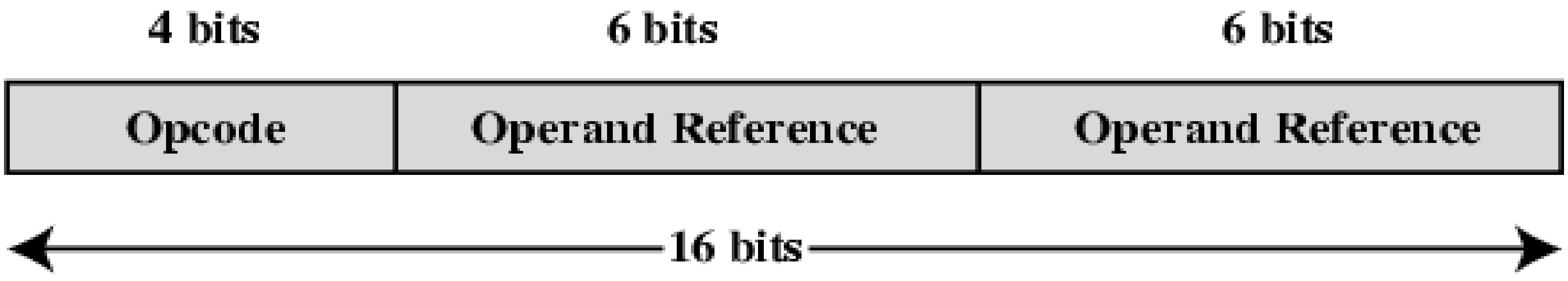
Lect 14: LC 3 Architecture

Instruction

- The instruction is the fundamental unit of work.
- Control Unit interprets instruction
- Specifies two things:
  - *opcode*: Operation to be performed (what the instruction does)
    - e.g. ADD, SUB, LOAD, STORE
  - *operands*: Data/locations to be used for operation
    - e.g ADD *dest scr1 scr2*
- An instruction is encoded as a sequence of bits.
- Often, but not always, instructions have a fixed length, such as 16 or 32 bits.
- ***Note:***
- A computer's instructions and their formats is known as its ***Instruction Set Architecture*** (***ISA***).

Elements of an Instruction

- Operation code (Op code)
  - Specifies the operation to be performed
- Source Operand reference
  - Operands that are input for the operation (one or more)
- Result Operand reference
  - Operands that stores the result of the operation
- Next Instruction Reference
  - Where to fetch the next instruction after this instruction



Instructions

- In LC-3
  - Each instruction consists of 16 bits
  numbered from left to right, bit [15] to bit [0].
- Bits [15:12] contains the opcode.
- Bits [11:0] are used to figure out where the operands are.
- There are at most 2^4 distinct opcodes.
- A computer's instructions and their formats is known as its Instruction Set Architecture (ISA).

Example: LC-3 ADD Instruction

- Each instruction has a four-bit opcode, bits [15:12].
  - LC – 3 can have maximum 16 different opcodes
- LC-3 has eight registers (R0-R7) for temporary storage.
  - Sources and destination of ADD are registers.
  - For ADD operation, at least one source operand (often both) should be in register

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD				Dst			Src1			0	0	0	Src2		

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	1	0

"Add the contents of R2 to the contents of R6,
and store the result in R6."

Example: LC-3 LDR Instruction

- Load instruction -- Reads data from memory
- *Base + offset mode:*
  - Add offset to base register -- result is memory address
  - Load from memory address into destination register

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

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

"Add the value 6 to the contents of R3 to form a memory address. Load the contents of that memory location to R2."

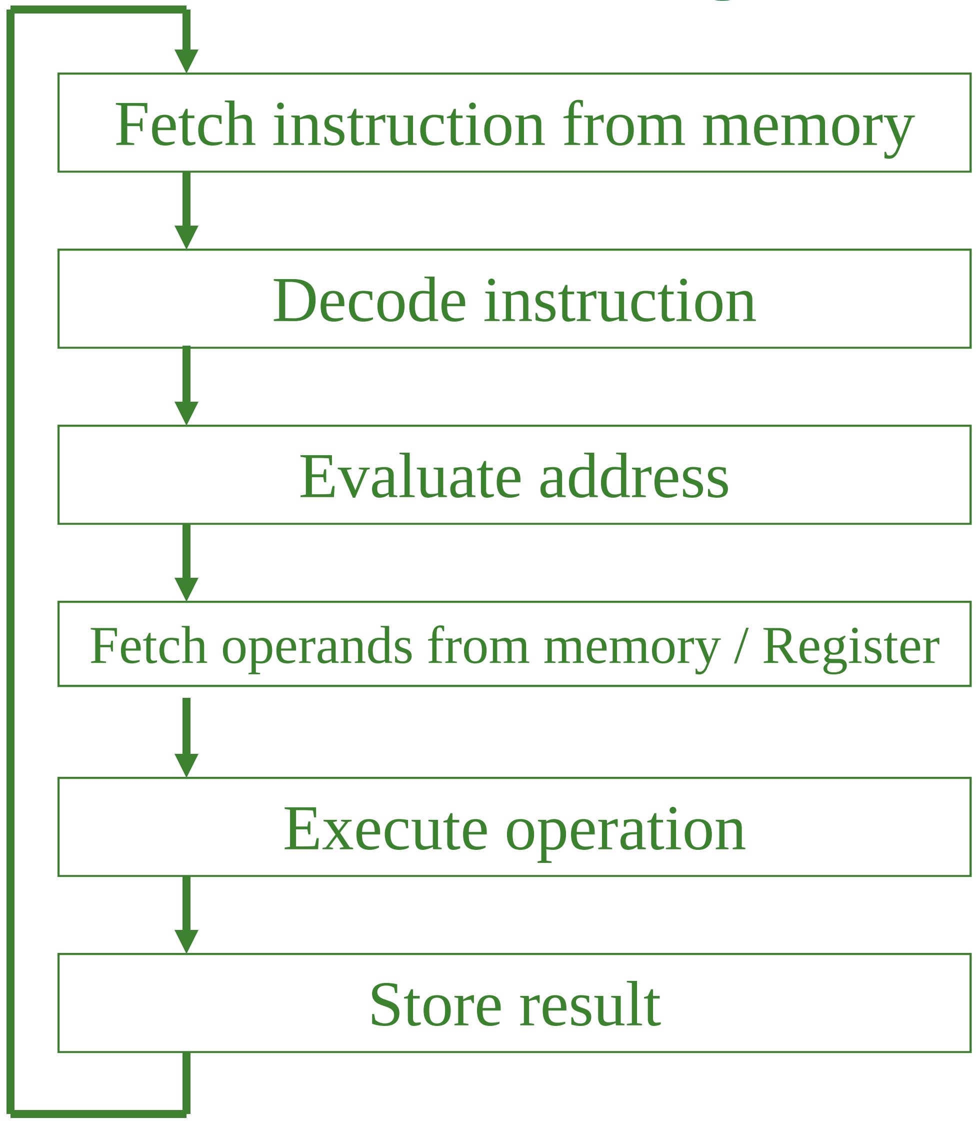
Instruction Processing

- Instructions look just like data –
  - it’s all interpretation.
- Three basic kinds of instructions:
  - Computational instructions (ADD, AND, …)
  - Data movement instructions (LD, ST, …)
  - Control instructions (JMP, BRnz, …)

Instruction Processing

- Instructions are processed under the direction of control unit in a systematic step by step manner.
- The sequence of steps is called the instruction cycle & each step is called a phase.
- Fundamentally, there are six phases to the instruction cycle.

Instruction Processing/Instruction Cycle

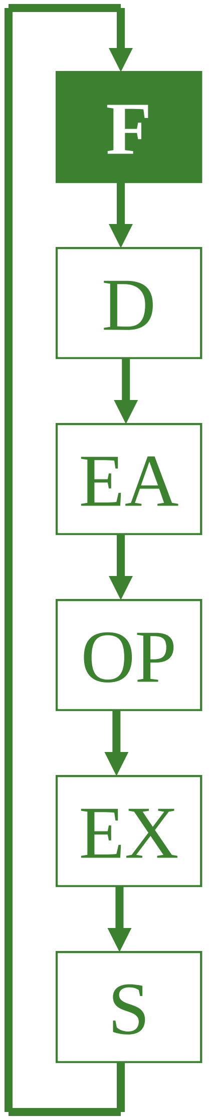

Fetch instruction from memory
Decode instruction
Evaluate address
Fetch operands from memory / Register
Execute operation
Store result

Instruction Processing/Instruction Cycle

- **Six phases of the complete Instruction Cycle**
  - **Fetch:** load IR with instruction from memory
  - **Decode:** determine action to take (set up inputs for ALU, RAM, etc.)
  - **Evaluate address:** compute memory address of operands, if any
  - **Fetch operands:** read operands from memory or registers
  - **Execute:** carry out instruction
  - **Store results:** write result to destination **(register or memory)**

Instruction Processing: FETCH

- Load next instruction from memory into Instruction Register (IR).
  - Copy (Load) contents of PC into MAR.
  - Send “read” signal to memory.
    - Results in copying the next instruction from memory into MDR.
  - Copy (Load) contents of MDR into IR.
- Then increment PC, so that it points to the next instruction in sequence.
  - PC should contain the address of the next instruction when fetch phase completes
  - PC becomes PC+1.



Instruction Cycle - step 1

- **Fetch**
  - **This actually takes several smaller steps, or "micro-instructions":**
    - **MAR ← (PC)** ; use the value in PC to access memory
    - **MDR ← Mem[MAR]** ; read memory location to MDR
    - **IR ← (MDR)** ; copy (MDR) to IR
    - **PC ← (PC)+1** ; Increment value of PC by 1

Actual Implementation in Hardware

- Step 1
  - Load MAR with the content of PC and Simultaneously increment the PC
- Step 2
  - Send “read signal” to memory, resulting in instruction being placed in the MDR
- Step 3
  - Load IR with the content of the MDR

Instruction Cycle - step 1

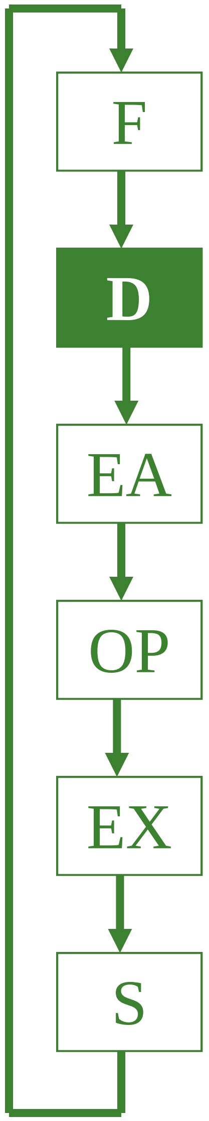
Fetch

- MAR ← (PC) ; use the value in PC to access memory
- PC ← (PC) + 1 ; increment the value of PC by 1
- MDR ← Mem[MAR] ; read memory location to MDR
- IR ← (MDR) ; copy (MDR) to IR

- Steps 1, 2 & 4 take a single machine cycle each,
- but step 3 (memory access) can take many machine cycles .

Instruction Processing: DECODE

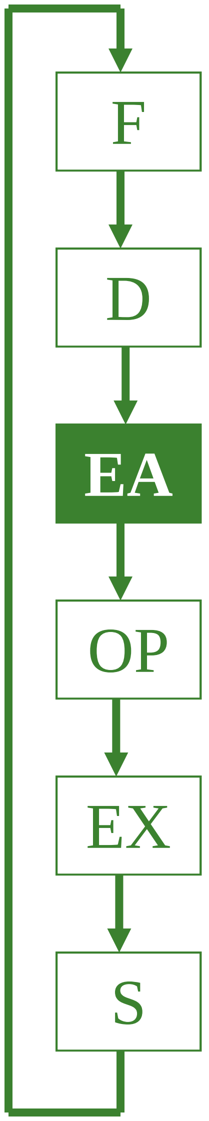
- First identify the opcode.
  - In LC-3, this is always the first four bits of instruction (IR[15:12]).
  - A 4-to-16 decoder asserts a control line corresponding to the desired opcode.
- Depending on opcode, identify what else is needed to process the instruction (operatnds etc..) from the remaining 12 - bits.
  - Example:
    - For LDR, last six bits is offset
    - For ADD, last three bits is source operand #2



Instruction Processing: EVALUATE ADDRESS

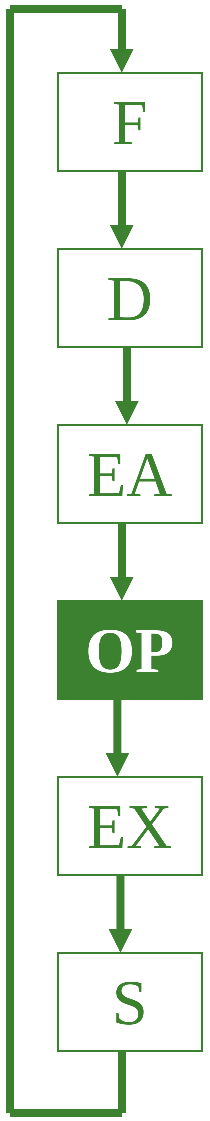
- For instructions that require memory access to get the operands; compute address used for access.

- Example:
  - Add offset to base register (as in LDR)
  - LDR R2, R1, #5
    - R2 → destination
    - R1 → Base
    - #5 → offset



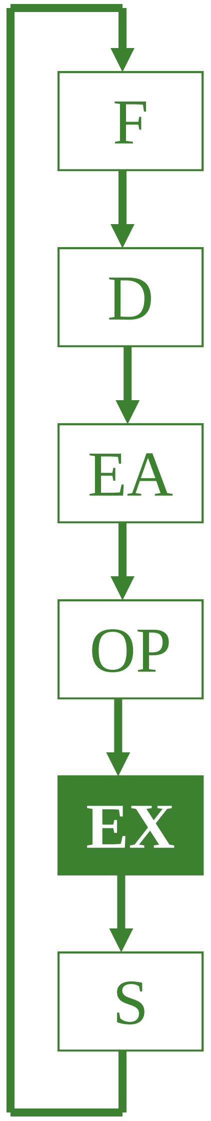
Instruction Processing: FETCH OPERANDS

- Obtain source operands needed to perform the operation.
- Operands can come from Registers or Memory or be embedded in the instruction itself.
- Examples:
  - In Load data from memory (LDR) FETCH takes 2 steps
    - loading MAR with the address calculated in the Effective Address phase.
    - reading memory (resulted in the source operand being placed in MDR)
  - Read data from register file (ADD)
    - FETCH phase consisted of obtaining the source operands from R2 & R6.



Instruction Processing: EXECUTE

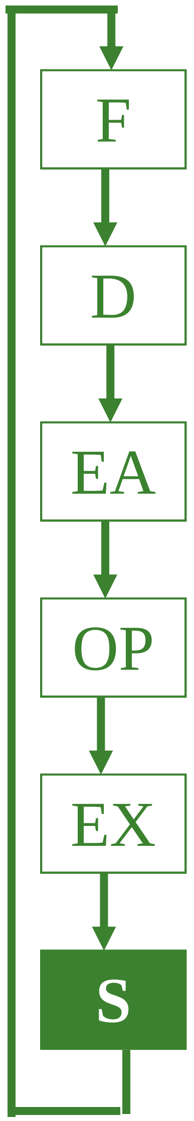
- Perform the operation, using the source operands.
- Examples:
  - For ADD
    - Sends operands to ALU and assert ADD signal
  - For Data movement instructions (load and store instructions)
    - Do nothing (i.e., No execution phase)



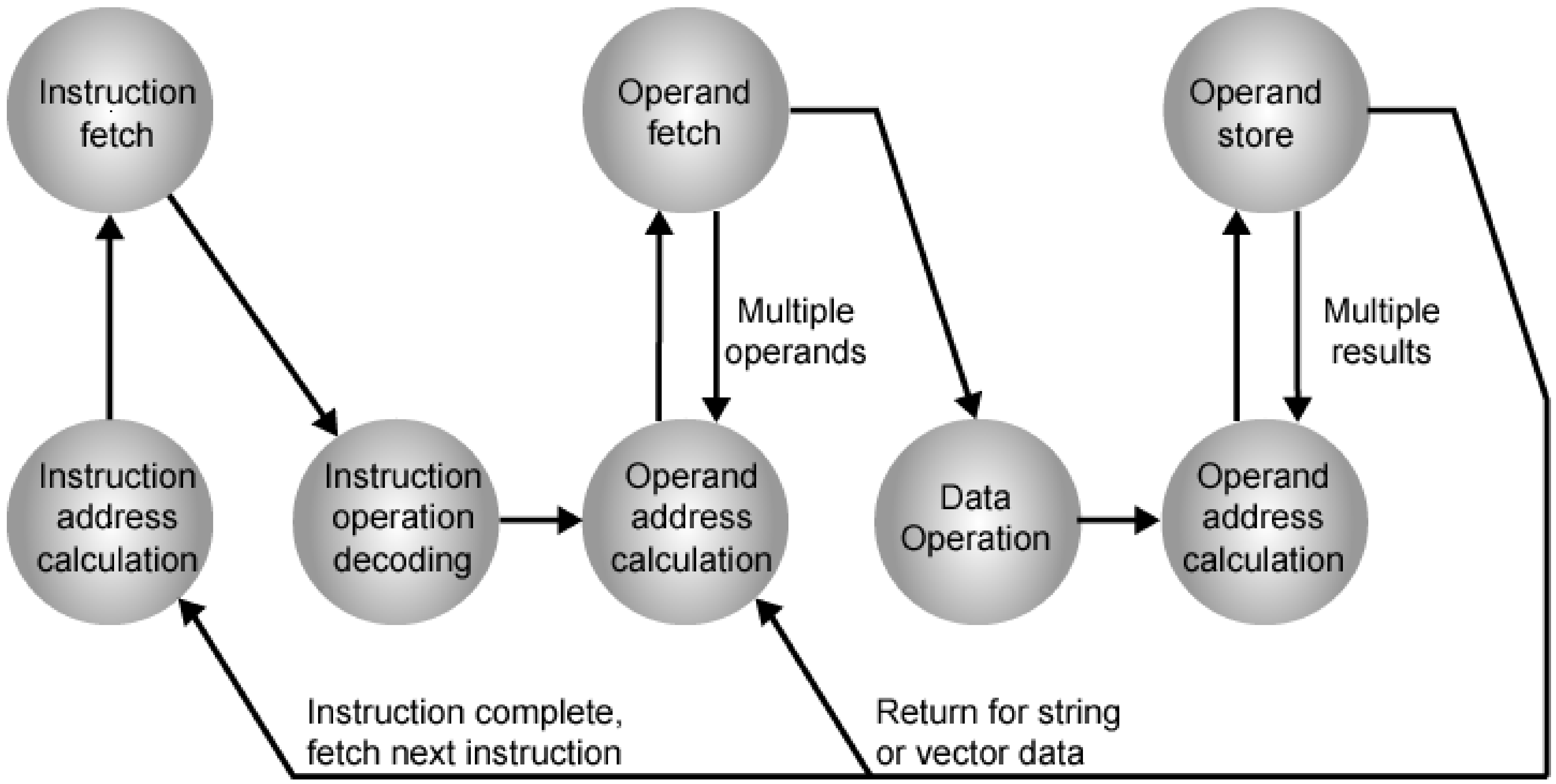
Instruction Processing: STORE RESULT

- Write results to destination. (Register or Memory)

- Examples:
  - Result of ADD is placed in destination register
  - Result of memory load is placed in destination register
  - For store instruction, data is stored to memory



Instruction Cycle State Diagram


Instruction fetch
Operand fetch
Operand store
Multiple operands
Multiple results
Instruction address calculation
Instruction operation decoding
Operand address calculation
Data Operation
Operand address calculation
Instruction complete, fetch next instruction
Return for string or vector data