+

Bahria University
Discovering Knowledge

Computer Architecture and Logic Design (CALD)

Lecture 06

Dr. Sorath Hansrajani

Assistant Professor

Department of Software Engineering

Bahria University Karachi Campus

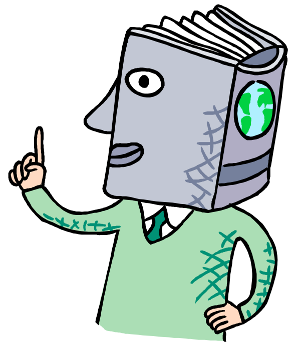
Email: sorathhansrajani.bukc@bahria.edu.pk

Instruction Sets: Characteristics and Functions

+

Machine Instruction Characteristics

- The operation of the processor is determined by the instructions it executes, referred to as *machine instructions* or *computer instructions*
- The collection of different instructions that the processor can execute is referred to as the processor's *instruction set*
- Each instruction must contain the information required by the processor for execution



Elements of a Machine Instruction

Operation code (opcode)

- Specifies the operation to be performed. The operation is specified by a binary code, known as the operation code, or *opcode*

Source operand reference

- The operation may involve one or more source operands, that is, operands that are inputs for the operation

Result operand reference

- The operation may produce a result

Next instruction reference

- This tells the processor where to fetch the next instruction after the execution of this instruction is complete

+
Instruction Cycle State Diagram

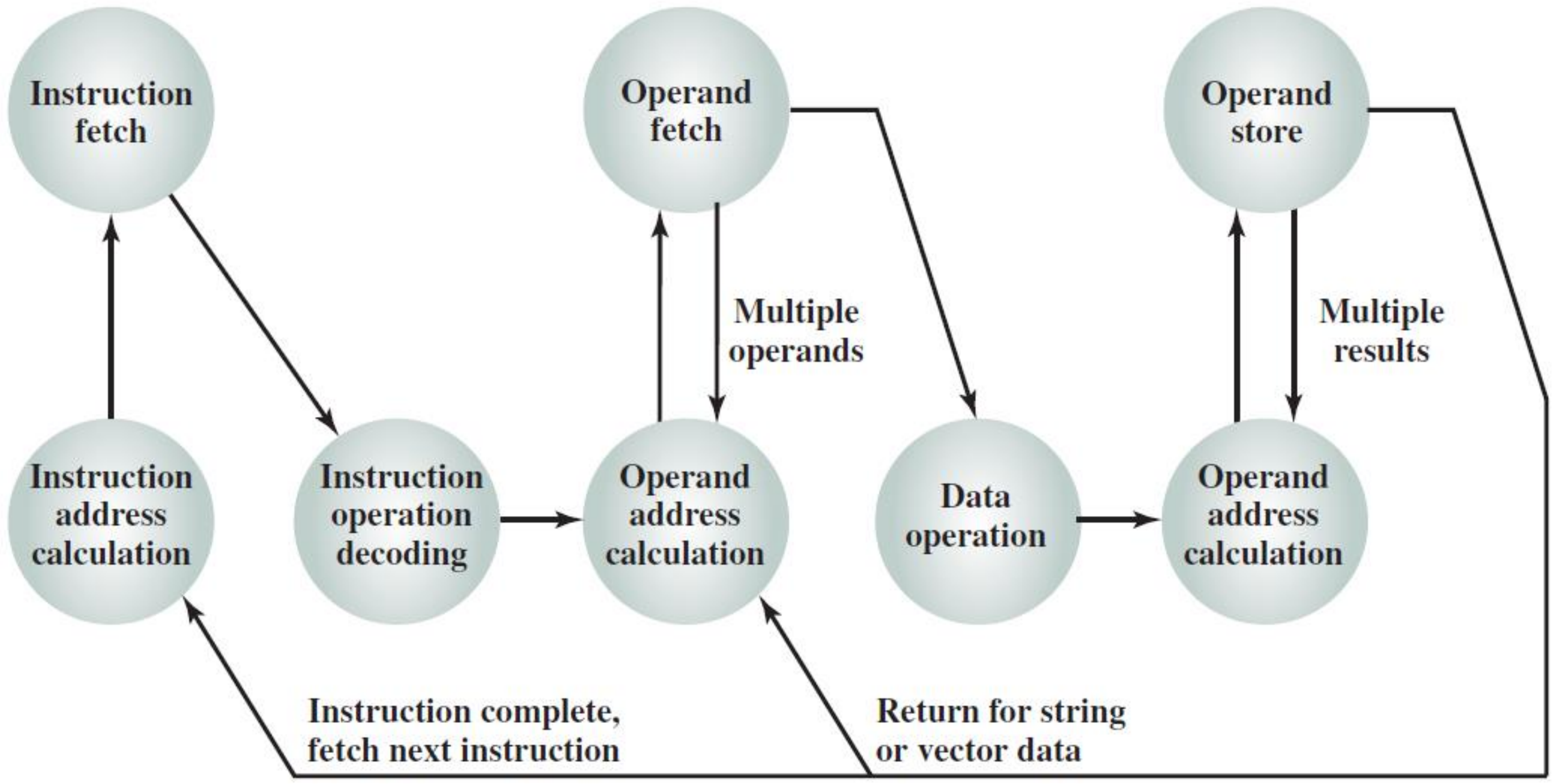

Instruction fetch
Operand fetch
Operand store
Multiple operands
Multiple results
Instruction address calculation
Instruction operation decoding
Operand address calculation
Data operation
Operand address calculation
Instruction complete, fetch next instruction
Return for string or vector data

Figure 12.1 Instruction Cycle State Diagram

Source and result operands can be in one of four areas:

1) Main or virtual memory
   - As with next instruction references, the main or virtual memory address must be supplied

2) I/O device
   - The instruction must specify the I/O module and device for the operation. If memory-mapped I/O is used, this is just another main or virtual memory address

3) Processor register
   - A processor contains one or more registers that may be referenced by machine instructions.
   - If more than one register exists each register is assigned a unique name or number and the instruction must contain the number of the desired register

4) Immediate
   - The value of the operand is contained in a field in the instruction being executed

Instruction Representation

- Within the computer each instruction is represented by a sequence of bits
- The instruction is divided into fields, corresponding to the constituent elements of the instruction

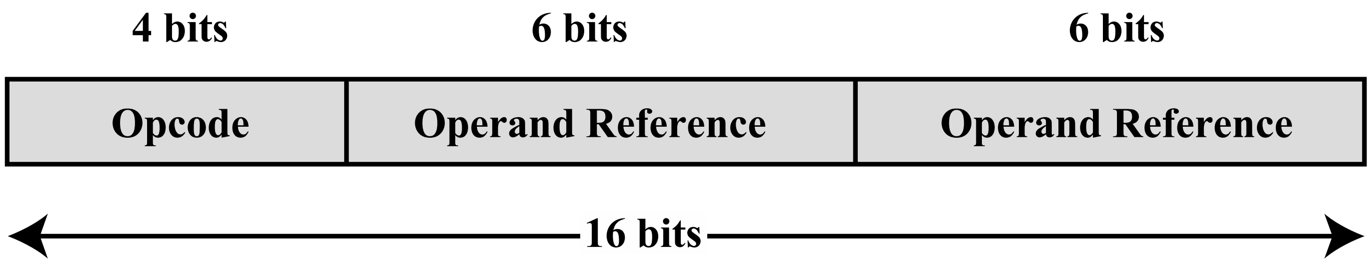


Figure 12.2 A Simple Instruction Format

Instruction Representation

- Opcodes are represented by abbreviations called *mnemonics*
- Examples include:

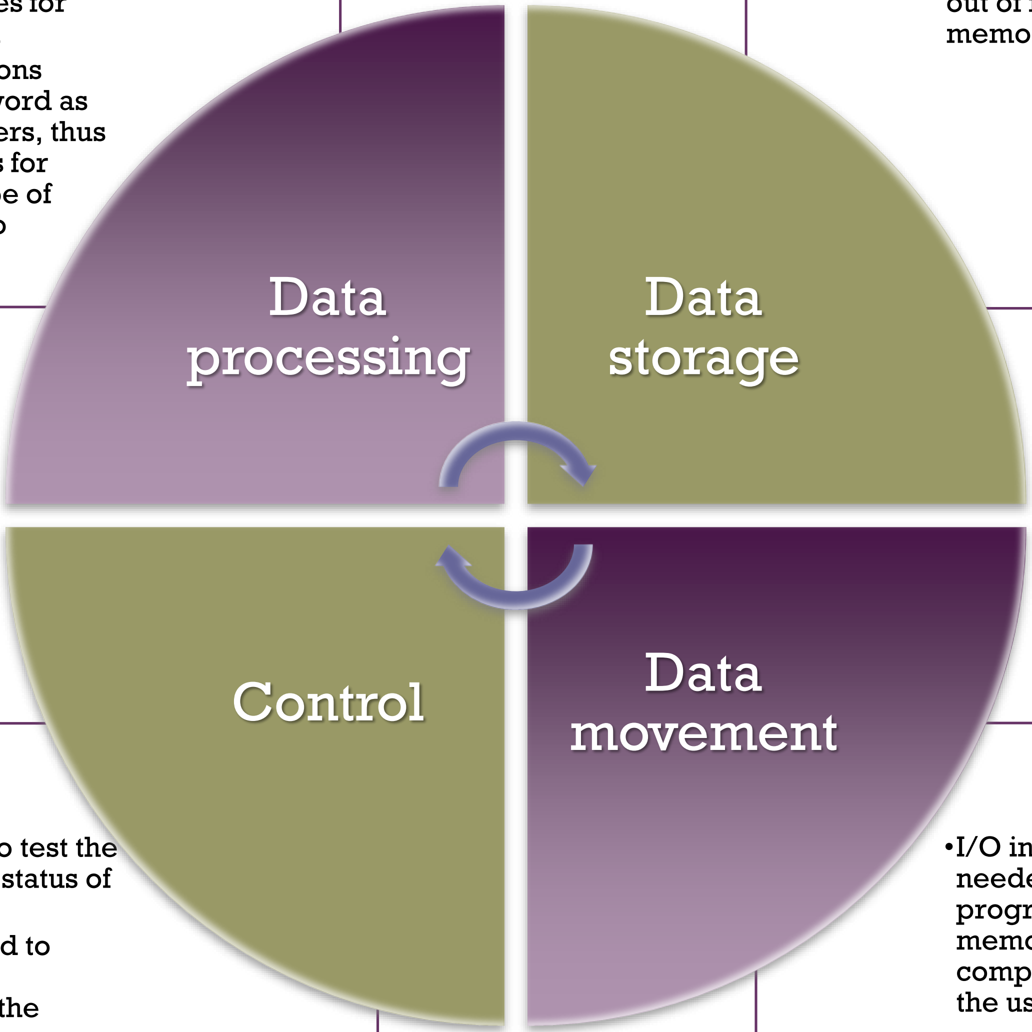
ADD	Add
SUB	Subtract
MUL	Multiply
DIV	Divide
LOAD	Load data from memory
STOR	Store data to memory

- Operands are also represented symbolically
- Each symbolic opcode has a fixed binary representation
  - The programmer specifies the location of each symbolic operand

Instruction Types

- Arithmetic instructions provide computational capabilities for processing numeric data
- Logic (Boolean) instructions operate on the bits of a word as bits rather than as numbers, thus they provide capabilities for processing any other type of data the user may wish to employ

- Movement of data into or out of register and or memory locations



- Test instructions are used to test the value of a data word or the status of a computation
- Branch instructions are used to branch to a different set of instructions depending on the decision made

- I/O instructions are needed to transfer programs and data into memory and the results of computations back out to the user

Instruction		Comment
SUB	Y, A, B	$Y \leftarrow A - B$
MPY	T, D, E	$T \leftarrow D \times E$
ADD	T, T, C	$T \leftarrow T + C$
DIV	Y, Y, T	$Y \leftarrow Y \div T$

(a) Three-address instructions

Instruction		Comment
MOVE	Y, A	$Y \leftarrow A$
SUB	Y, B	$Y \leftarrow Y - B$
MOVE	T, D	$T \leftarrow D$
MPY	T, E	$T \leftarrow T \times E$
ADD	T, C	$T \leftarrow T + C$
DIV	Y, T	$Y \leftarrow Y \div T$

(b) Two-address instructions

Instruction		Comment
LOAD	D	$AC \leftarrow D$
MPY	E	$AC \leftarrow AC \times E$
ADD	C	$AC \leftarrow AC + C$
STOR	Y	$Y \leftarrow AC$
LOAD	A	$AC \leftarrow A$
SUB	B	$AC \leftarrow AC - B$
DIV	Y	$AC \leftarrow AC \div Y$
STOR	Y	$Y \leftarrow AC$

(c) One-address instructions

Figure 12.3 Programs to Execute $Y = \dfrac{A - B}{C + (D \times E)}$

Table 12.1 Utilization of Instruction Addresses (Nonbranching Instructions)

Number of Addresses	Symbolic Representation	Interpretation
3	OP A, B, C	A ← B OP C
2	OP A, B	A ← A OP B
1	OP A	AC ← AC OP A
0	OP	T ← (T − 1) OP T

AC = accumulator
T = top of stack
(T − 1) = second element of stack
A, B, C = memory or register locations

+

Number of Addresses

- 3 addresses
  - Operand 1, Operand 2, Result
  - a = b + c;
  - May be a forth - next instruction (usually implicit)
  - Not common scheme
  - Needs very long words to hold everything

+

Number of Addresses

- 2 addresses
  - One address doubles as operand and result
  - a = a + b
  - Reduces length of instruction
  - Common format in most architectures
  - Requires some extra work
    - Temporary storage to hold some results

Number of Addresses

- 1 address
  - Implicit second address
  - Usually a register (accumulator)
  - Common on early machines

+

Number of Addresses

- 0 (zero) addresses
  - All addresses implicit
  - Uses a stack
  - e.g. push a
  -       push b
  -       add
  -       pop c

  - c = a + b

+

Instruction Addresses

- The number of addresses per instruction is a basic design decision.
- Fewer addresses per instruction:
  - less complex processor.
  - instructions of shorter length.
  - programs contain more total instructions
  - results in longer execution times and
  - longer, more complex programs.

+

Instruction Addresses

- Also, there is an important threshold between one-address and multiple-address instructions.
- With one-address instructions, the programmer generally has available only one general-purpose register, the accumulator.
- With multiple-address instructions, it is common to have multiple general-purpose registers.
- This allows some operations to be performed solely on registers.
- Because register references are faster than memory references, this speeds up execution.
- For reasons of flexibility and ability to use multiple registers, most contemporary machines employ a mixture of two- and three-address instructions.

Instruction Set Design

Very complex because it affects so many aspects of the computer system

↓

Defines many of the functions performed by the processor

↓

Programmer's means of controlling the processor

↓

Fundamental design issues:

Operation repertoire	Data types	Instruction format	Registers	Addressing
• How many and which operations to provide and how complex operations should be	• The various types of data upon which operations are performed	• Instruction length in bits, number of addresses, size of various fields, etc.	• Number of processor registers that can be referenced by instructions and their use	• The mode or modes by which the address of an operand is specified

+

Fundamental Design Issues

- Operation repertoire
  - How many and which operations to provide, and how complex operations should be
- Data types
  - The various types of data upon which operations are performed
- Instruction format
  - Instruction length (in bits), number of addresses, size of various fields, and so on
- Registers
  - Number of processor registers that can be referenced by instructions, and their use
- Addressing
  - The mode or modes by which the address of an operand is specified

Types of Operands

Addresses

Numbers
Integers, floating point

Characters
ASCII

Logical Data
Bits or flags

Types of Operations

- Data transfer
- Arithmetic
- Logical
- Conversion
- I/O
- System control
- Transfer of control

Table 12.3 Common Instruction Set Operations

Type	Operation Name	Description
Data transfer	Move (transfer)	Transfer word or block from source to destination
	Store	Transfer word from processor to memory
	Load (fetch)	Transfer word from memory to processor
	Exchange	Swap contents of source and destination
	Clear (reset)	Transfer word of 0s to destination
	Set	Transfer word of 1s to destination
	Push	Transfer word from source to top of stack
	Pop	Transfer word from top of stack to destination
Arithmetic	Add	Compute sum of two operands
	Subtract	Compute difference of two operands
	Multiply	Compute product of two operands
	Divide	Compute quotient of two operands
	Absolute	Replace operand by its absolute value
	Negate	Change sign of operand
	Increment	Add 1 to operand
	Decrement	Subtract 1 from operand
Logical	AND	Perform logical AND
	OR	Perform logical OR
	NOT	(complement) Perform logical NOT
	Exclusive-OR	Perform logical XOR
	Test	Test specified condition; set flag(s) based on outcome
	Compare	Make logical or arithmetic comparison of two or more operands; set flag(s) based on outcome
	Set Control Variables	Class of instructions to set controls for protection purposes, interrupt handling, timer control, etc.
	Shift	Left (right) shift operand, introducing constants at end
	Rotate	Left (right) shift operand, with wraparound end

Type	Operation Name	Description
Input/output	Input (read)	Transfer data from specified I/O port or device to destination (e.g., main memory or processor register)
	Output (write)	Transfer data from specified source to I/O port or device
	Start I/O	Transfer instructions to I/O processor to initiate I/O operation
	Test I/O	Transfer status information from I/O system to specified destination
Conversion	Translate	Translate values in a section of memory based on a table of correspondences
	Convert	Convert the contents of a word from one form to another (e.g., packed decimal to binary)
Transfer of control	Jump (branch)	Unconditional transfer; load PC with specified address
	Jump Conditional	Test specified condition; either load PC with specified address or do nothing, based on condition
	Jump to Subroutine	Place current program control information in known location; jump to specified address
	Return	Replace contents of PC and other register from known location
	Execute	Fetch operand from specified location and execute as instruction; do not modify PC
	Skip	Increment PC to skip next instruction
	Skip Conditional	Test specified condition; either skip or do nothing based on condition
	Halt	Stop program execution
	Wait (hold)	Stop program execution; test specified condition repeatedly; resume execution when condition is satisfied
	No operation	No operation is performed, but program execution is continued

Processor Actions

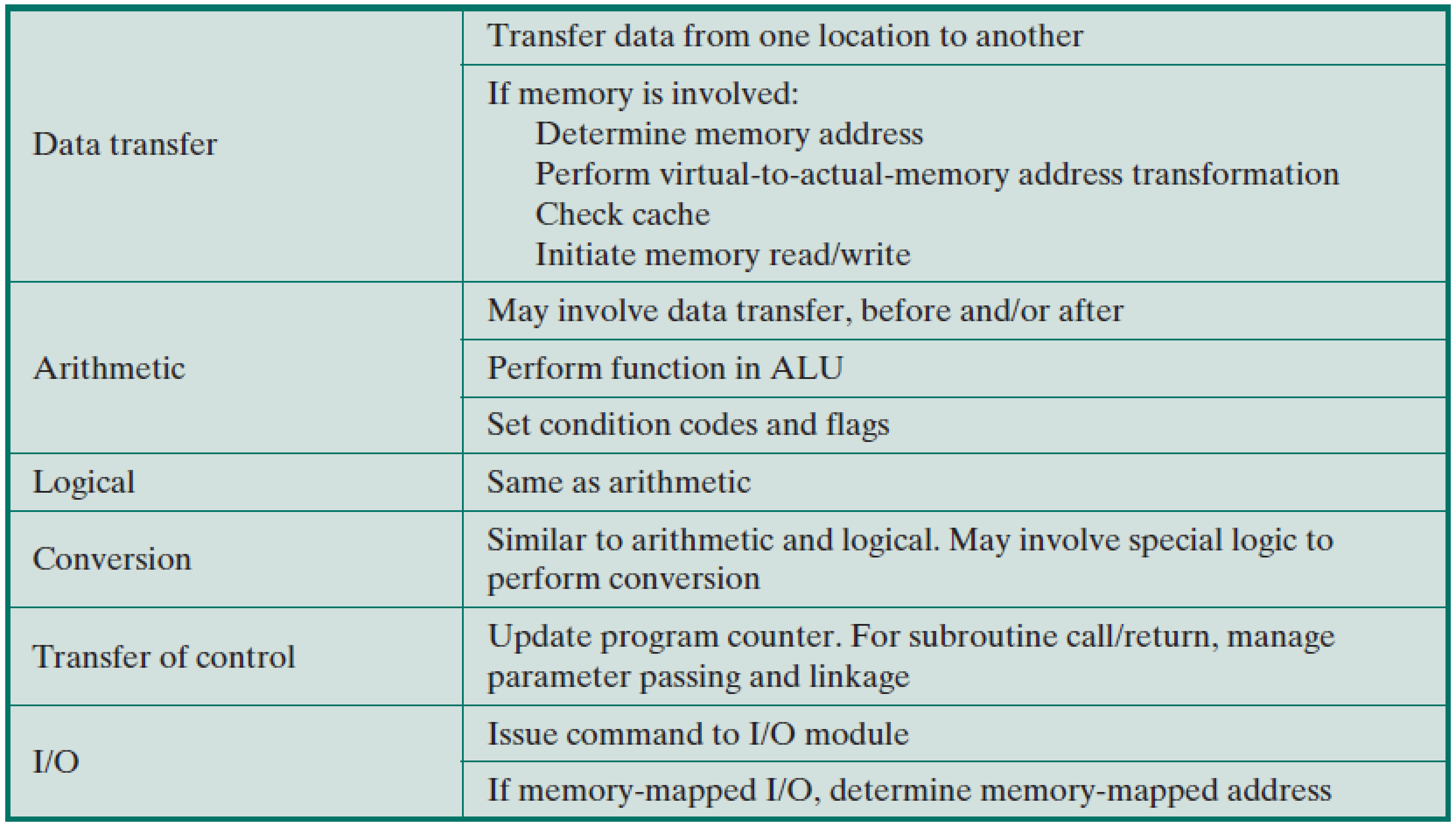


Table 12.4 Processor Actions for Various Types of Operations

Data transfer	Transfer data from one location to another
	If memory is involved: Determine memory address Perform virtual-to-actual-memory address transformation Check cache Initiate memory read/write
Arithmetic	May involve data transfer, before and/or after
	Perform function in ALU
	Set condition codes and flags
Logical	Same as arithmetic
Conversion	Similar to arithmetic and logical. May involve special logic to perform conversion
Transfer of control	Update program counter. For subroutine call/return, manage parameter passing and linkage
I/O	Issue command to I/O module
	If memory-mapped I/O, determine memory-mapped address

Data Transfer

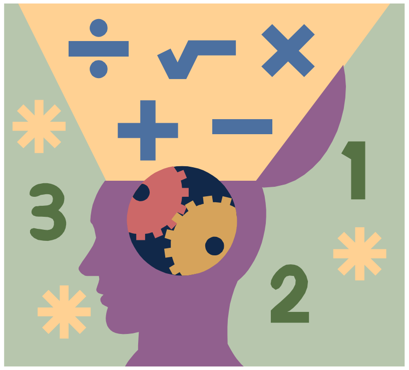
Most fundamental type of machine instruction

Must specify:

- Location of the source and destination operands
- The length of data to be transferred must be indicated
- The mode of addressing for each operand must be specified

+

Arithmetic



- Add, Subtract, Multiply, Divide
- Signed Integer
- Floating point ?
- May include
  - Increment (a++)
  - Decrement (a--)
  - Negate (-a)

+

Logical

Table 12.6 Basic Logical Operations

P	Q	NOT P	P AND Q	P OR Q	P XOR Q	P = Q
0	0	1	0	0	0	1
0	1	1	0	1	1	0
1	0	0	0	1	1	0
1	1	0	1	1	0	1

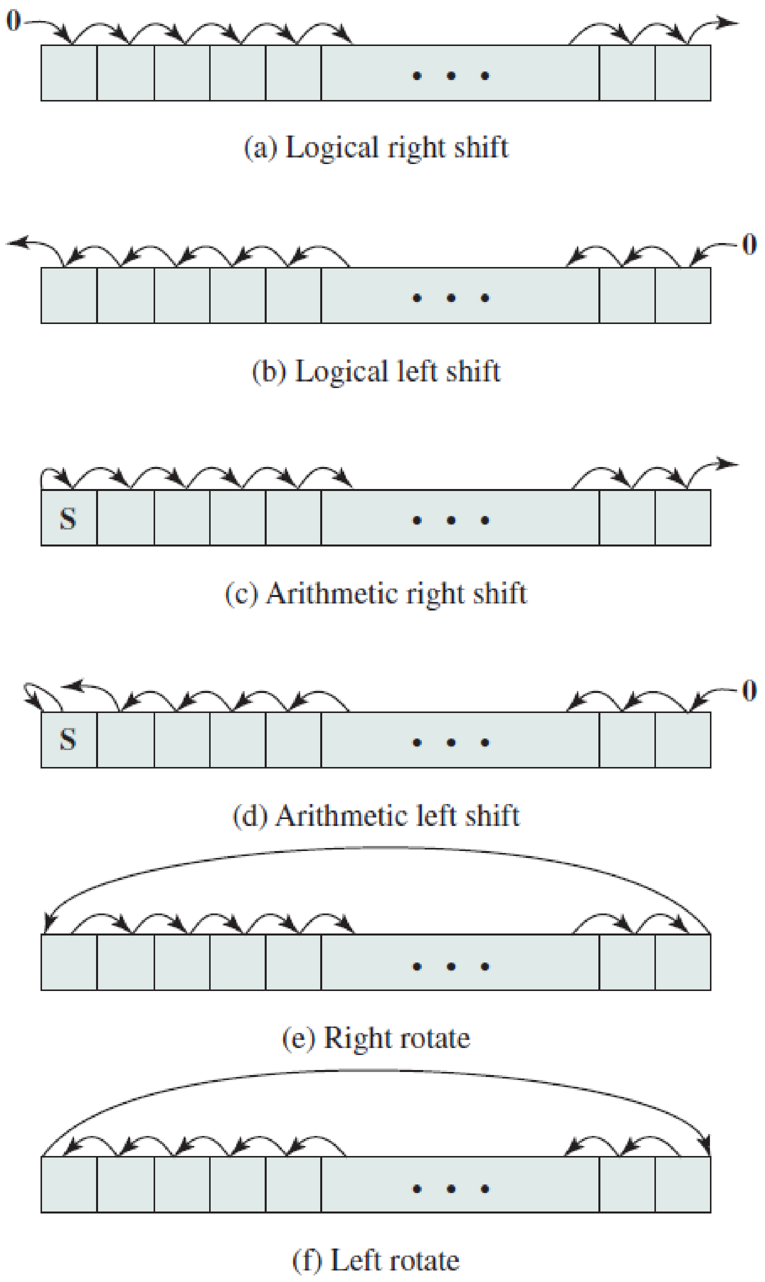
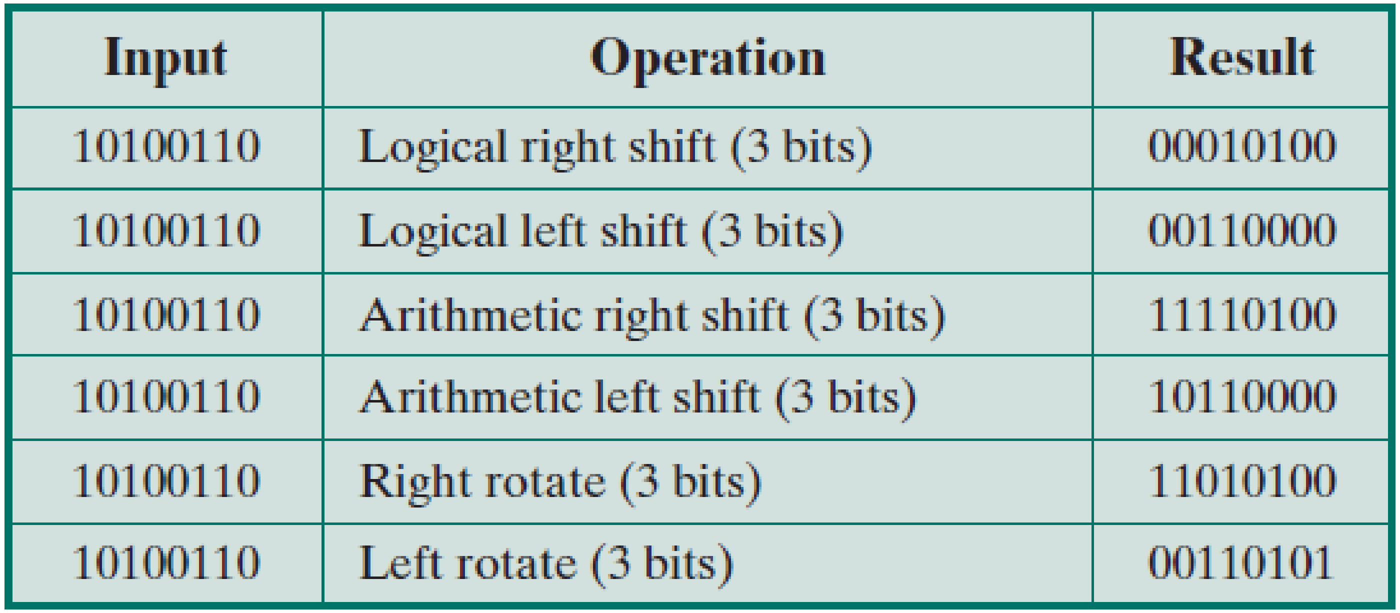


Figure 12.6 Shift and Rotate Operations

+

Logical

Table 12.7 Examples of Shift and Rotate Operations



Input	Operation	Result
10100110	Logical right shift (3 bits)	00010100
10100110	Logical left shift (3 bits)	00110000
10100110	Arithmetic right shift (3 bits)	11110100
10100110	Arithmetic left shift (3 bits)	10110000
10100110	Right rotate (3 bits)	11010100
10100110	Left rotate (3 bits)	00110101

Conversion

Instructions that change the format or operate on the format of data

An example is converting from decimal to binary

An example of a more complex editing instruction is the EAS/390 Translate (TR) instruction

+

Input/Output

- Variety of approaches taken:
  - Isolated programmed I/O
  - Memory-mapped programmed I/O
  - DMA
  - Use of an I/O processor
- Many implementations provide only a few I/O instructions, with the specific actions specified by parameters, codes, or command words

System Control

Instructions that can be executed only while the processor is in a certain privileged state or is executing a program in a special privileged area of memory

Typically these instructions are reserved for the use of the operating system

Examples of system control operations:

- A system control instruction may read or alter a control register
- An instruction to read or modify a storage protection key
- Access to process control blocks in a multiprogramming system

+
Transfer of Control

- Reasons why transfer-of-control operations are required:
  - It is essential to be able to execute each instruction more than once
  - Virtually all programs involve some decision making
  - It helps if there are mechanisms for breaking the task up into smaller pieces that can be worked on one at a time
- Most common transfer-of-control operations found in instruction sets:
  - Branch
  - Skip
  - Procedure call

+

Transfer of Control (Branch Instructions)

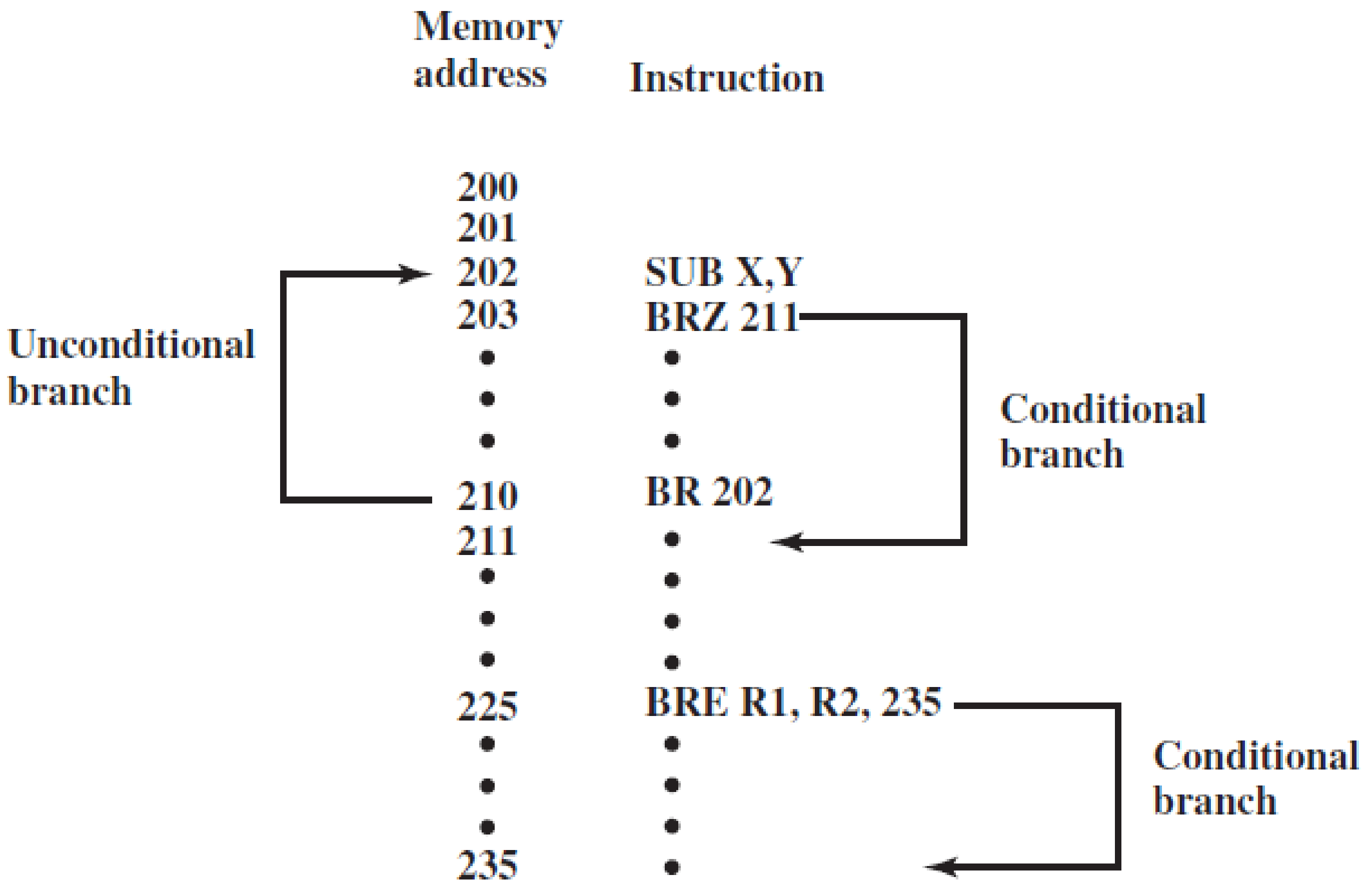


Figure 12.7 Branch Instructions

Transfer of Control (Skip Instructions)

Includes an implied address

Typically implies that one instruction be skipped, thus the implied address equals the address of the next instruction plus one instruction length

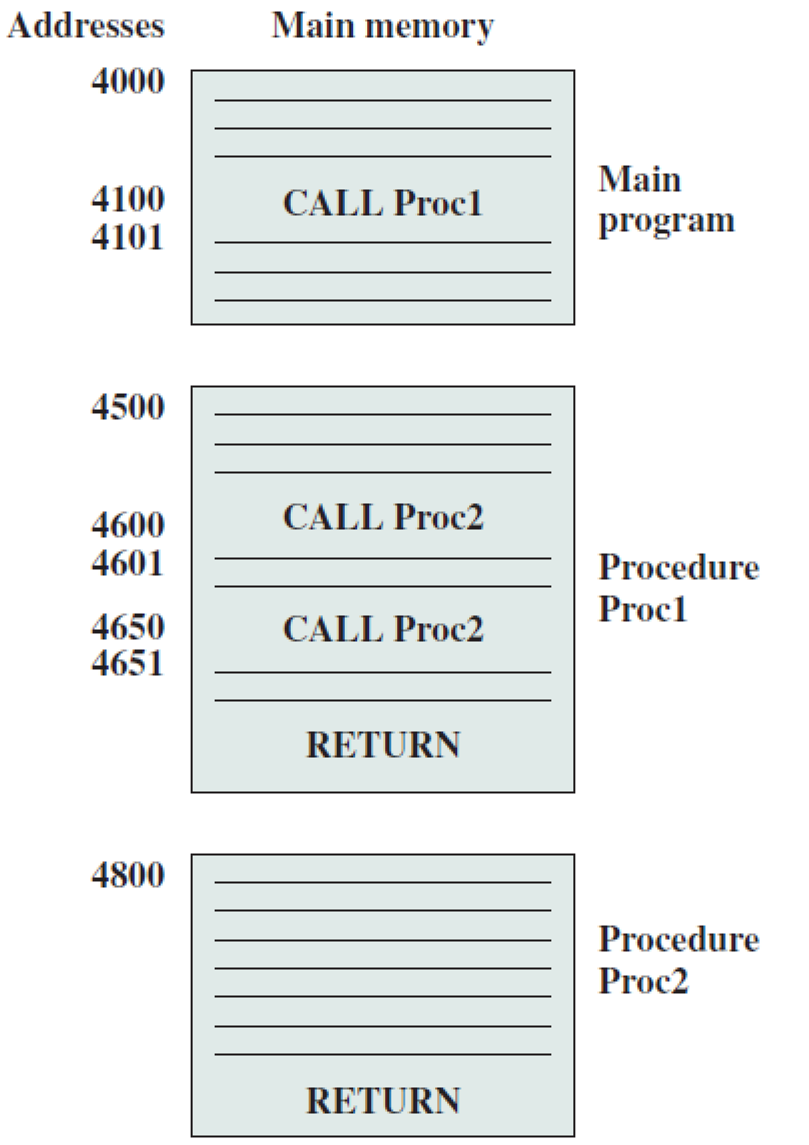
Because the skip instruction does not require a destination address field it is free to do other things

Example is the increment-and-skip-if-zero (ISZ) instruction

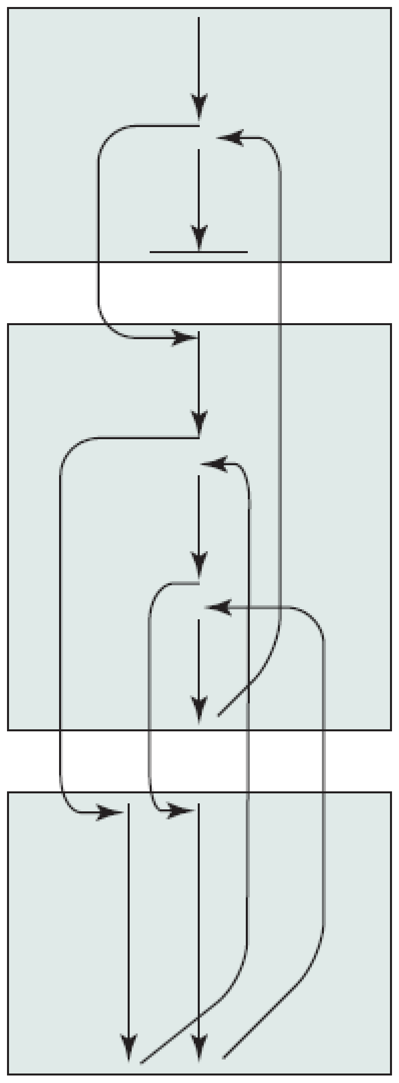
+

Transfer of Control (Procedure Call Instructions)

- Self-contained computer program that is incorporated into a larger program
  - At any point in the program the procedure may be invoked, or *called*
  - Processor is instructed to go and execute the entire procedure and then return to the point from which the call took place
- Two principal reasons for use of procedures:
  - Economy
    - A procedure allows the same piece of code to be used many times
  - Modularity
- Involves two basic instructions:
  - A call instruction that branches from the present location to the procedure
  - Return instruction that returns from the procedure to the place from which it was called



(a) Calls and returns



(b) Execution sequence

Figure 12.8 Nested Procedures

Transfer of Control (Procedure Call Instructions)

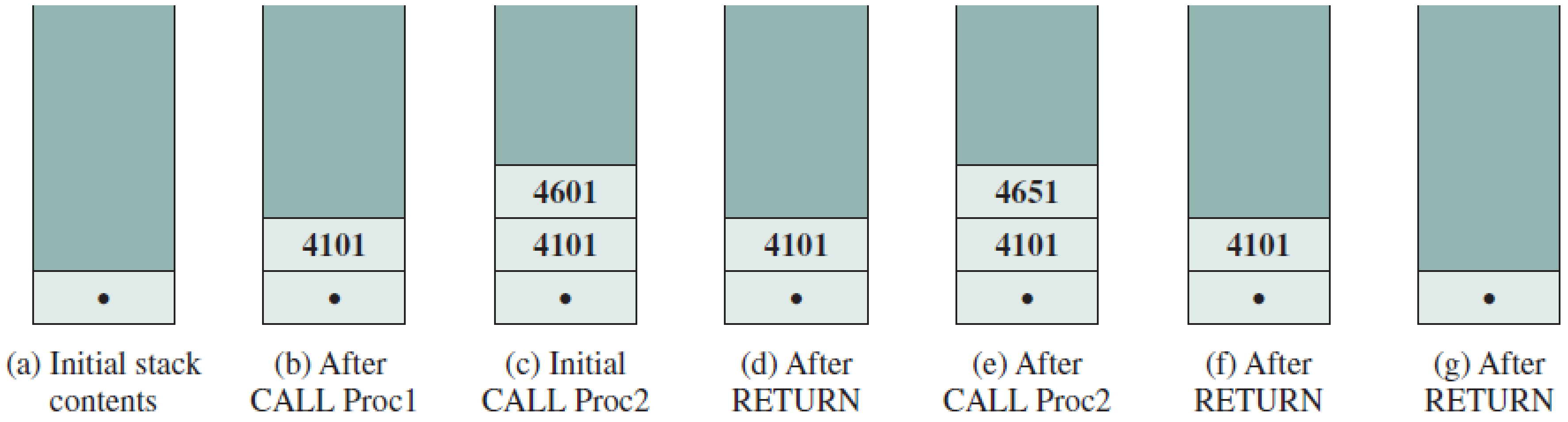


Figure 12.9 Use of Stack to Implement Nested Subroutines of Figure 12.8

ARM Operation Types

- Load and store instructions
- Branch instructions
- Data-processing instructions
- Multiply instructions
- Parallel addition and subtraction instructions
- Extend instructions
- Status register access instructions

Table 12.11 ARM Conditions for Conditional Instruction Execution

Code	Symbol	Condition Tested	Comment
0000	EQ	Z = 1	Equal
0001	NE	Z = 0	Not equal
0010	CS/HS	C = 1	Carry set/unsigned higher or same
0011	CC/LO	C = 0	Carry clear/unsigned lower
0100	MI	N = 1	Minus/negative
0101	PL	N = 0	Plus/positive or zero
0110	VS	V = 1	Overflow
0111	VC	V = 0	No overflow
1000	HI	C = 1 AND Z = 0	Unsigned higher
1001	LS	C = 0 OR Z = 1	Unsigned lower or same
1010	GE	N = V [(N = 1 AND V = 1) OR (N = 0 AND V = 0)]	Signed greater than or equal
1011	LT	N ≠ V [(N = 1 AND V = 0) OR (N = 0 AND V = 1)]	Signed less than
1100	GT	(Z = 0) AND (N = V)	Signed greater than
1101	LE	(Z = 1) OR (N ≠ V)	Signed less than or equal
1110	AL	—	Always (unconditional)
1111	—	—	This instruction can only be executed unconditionally

+

Summary

Chapter 12

Instruction Sets: Characteristics and Functions

- Machine instruction characteristics
  - Elements of a machine instruction
  - Instruction representation
  - Instruction types
  - Number of addresses
  - Instruction set design
- Types of operands
  - Numbers
  - Characters
  - Logical data
- Types of operations
  - Data transfer
  - Arithmetic
  - Logical
  - Conversion
  - Input/output
  - System control
  - Transfer of control
- ARM operation types