INSTRUCTION SETS: CHARACTERISTICS AND FUNCTIONS

MACHINE INSTRUCTION CHARACTERISTICS

- The operation of the processor is determined by the instructions it executes referred 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*.

Elements of a Machine Instruction

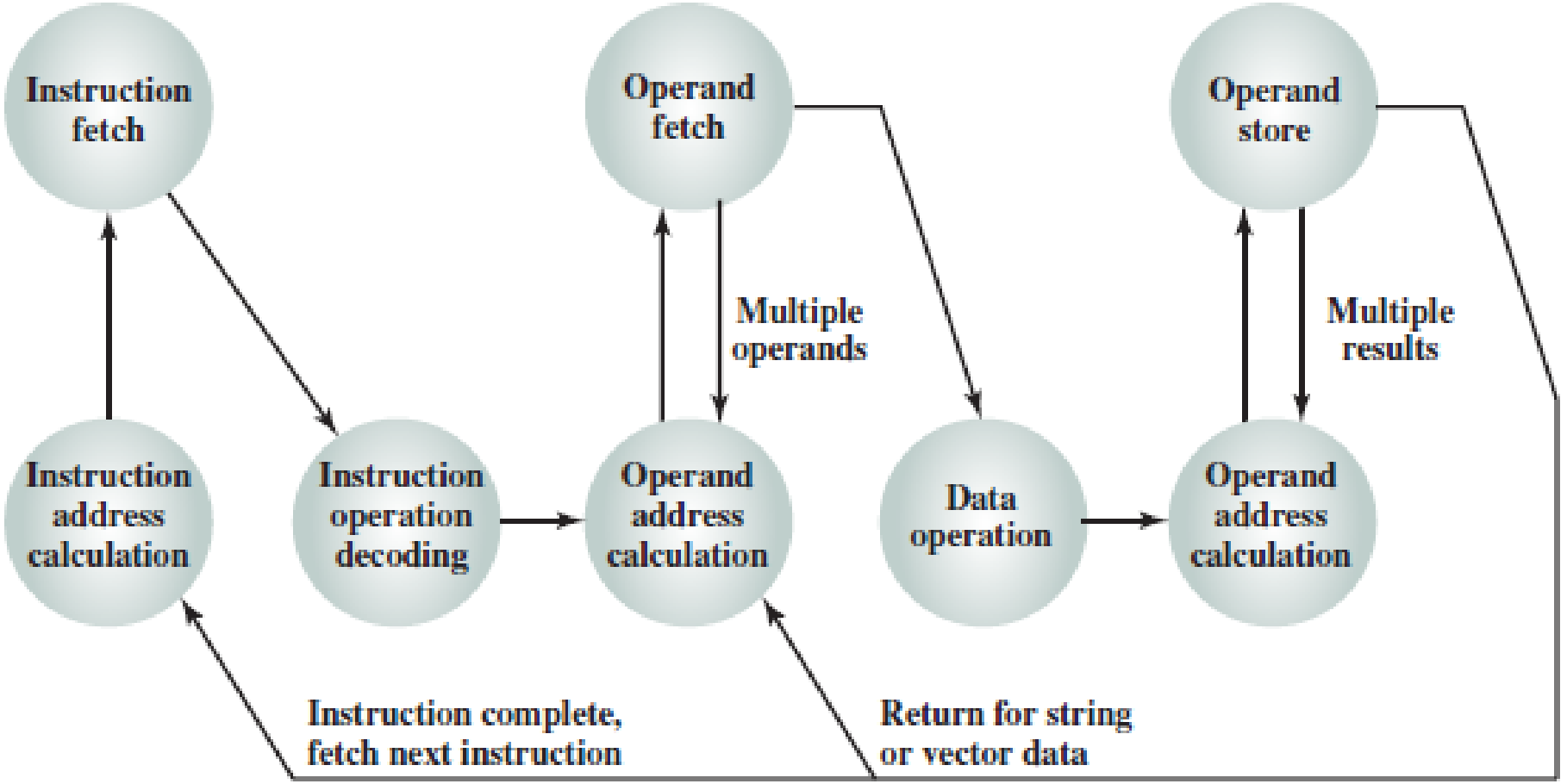


Figure 12.1 Instruction Cycle State Diagram

Elements of a Machine Instruction

- **Operation code:** Specifies the operation to be performed (e.g., ADD, I/O). 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.

Elements of a Machine Instruction

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

- **Main or virtual memory:** As with next instruction references, the main or virtual memory address must be supplied.
- **Processor register:** With rare exceptions, a processor contains one or more registers that may be referenced by machine instructions. If only one register exists, reference to it may be implicit. If more than one register exists, then each register is assigned a unique name or number, and the instruction must contain the number of the desired register.
- **Immediate:** The value of the operand is contained in a field in the instruction being executed.
- **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.

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.

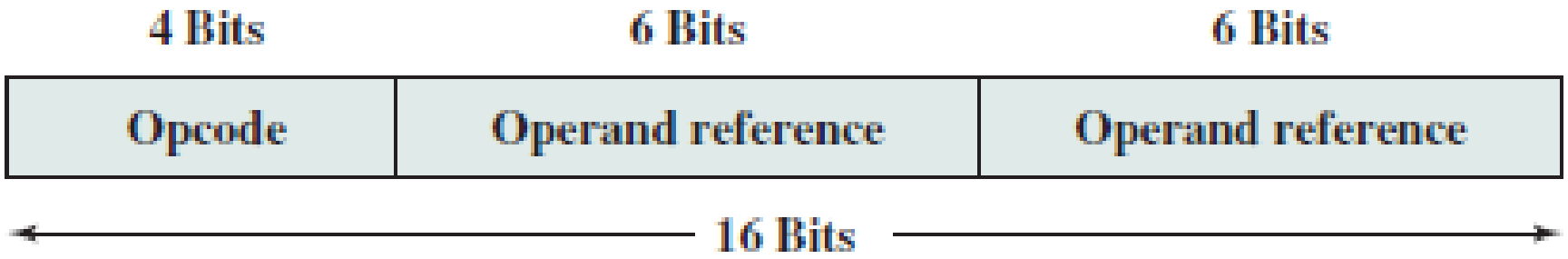


Figure 12.2 A Simple Instruction Format

Opcodes

- Opcodes are represented by abbreviations, called mnemonics, that indicate the operation. Common 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. For example, the instruction

  ADD R, Y

The IAS Instruction

Instruction Type	Opcode	Symbolic Representation	Description
Data transfer	00001010	LOAD MQ	Transfer contents of register MQ to the accumulator AC
	00001001	LOAD MQ,M(X)	Transfer contents of memory location X to MQ
	00100001	STOR M(X)	Transfer contents of accumulator to memory location X
	00000001	LOAD M(X)	Transfer M(X) to the accumulator
	00000010	LOAD –M(X)	Transfer –M(X) to the accumulator
	00000011	LOAD |M(X)|	Transfer absolute value of M(X) to the accumulator
	00000100	LOAD –|M(X)|	Transfer –|M(X)| to the accumulator
Unconditional branch	00001101	JUMP M(X,0:19)	Take next instruction from left half of M(X)
	00001110	JUMP M(X,20:39)	Take next instruction from right half of M(X)
Conditional branch	00001111	JUMP + M(X,0:19)	If number in the accumulator is nonnegative, take next instruction from left half of M(X)
	00010000	JUMP + M(X,20:39)	If number in the accumulator is nonnegative, take next instruction from right half of M(X)
Arithmetic	00000101	ADD M(X)	Add M(X) to AC; put the result in AC
	00000111	ADD |M(X)|	Add |M(X)| to AC; put the result in AC
	00000110	SUB M(X)	Subtract M(X) from AC; put the result in AC
	00001000	SUB |M(X)|	Subtract |M(X)| from AC; put the remainder in AC
	00001011	MUL M(X)	Multiply M(X) by MQ; put most significant bits of result in AC, put least significant bits in MQ
	00001100	DIV M(X)	Divide AC by M(X); put the quotient in MQ and the remainder in AC
	00010100	LSH	Multiply accumulator by 2; that is, shift left one bit position
	00010101	RSH	Divide accumulator by 2; that is, shift right one position
Address modify	00010010	STOR M(X,8:19)	Replace left address field at M(X) by 12 rightmost bits of AC
	00010011	STOR M(X,28:39)	Replace right address field at M(X) by 12 rightmost bits

Instruction Types

- X = X + Y

  1. Load a register with the contents of memory location 513.

  2. Add the contents of memory location 514 to the register.

  3. Store the contents of the register in memory location 513.

- The single high level instruction may require three machine instructions. This is typical of the relationship between a high- level language and a machine language.

Instruction Types

- We can categorize instruction types as follows:
  - **Data processing:** Arithmetic and logic instructions.
  - **Data storage:** Movement of data into or out of register and or memory locations.
  - **Data movement:** I/O instructions.
  - **Control:** Test and branch instructions.

Number of Addresses

- What is the maximum number of addresses one might need in an instruction?
- Arithmetic and logic instructions will require the most operands. Virtually all arithmetic and logic operations are either unary (one source operand) or binary (two source operands). Thus, we would need a maximum of two addresses to reference source operands.
- The result of an operation must be stored, suggesting a third address, which defines a destination operand.
- Finally, after completion of an instruction, the next instruction must be fetched, and its address is needed.
- In most architectures, many instructions have one, two, or three operand addresses, with the address of the next instruction being implicit (obtained from the program counter).

Number of Addresses

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 $Y = \dfrac{A - B}{C + (D \times E)}$

Number of Addresses

- Three-address instruction formats are not common because they require a relatively long instruction format to hold the three address references.
- With two-address instructions, and for binary operations, one address must do double duty as both an operand and a result.
- In one-address instruction. For this to work, a second address must be implicit. This was common in earlier machines, with the implied address being a processor register known as the **accumulator** (AC).

Instruction Set Design

- The instruction set defines many of the functions performed by the processor and thus has a significant effect on the implementation of the processor.
- The most important 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.

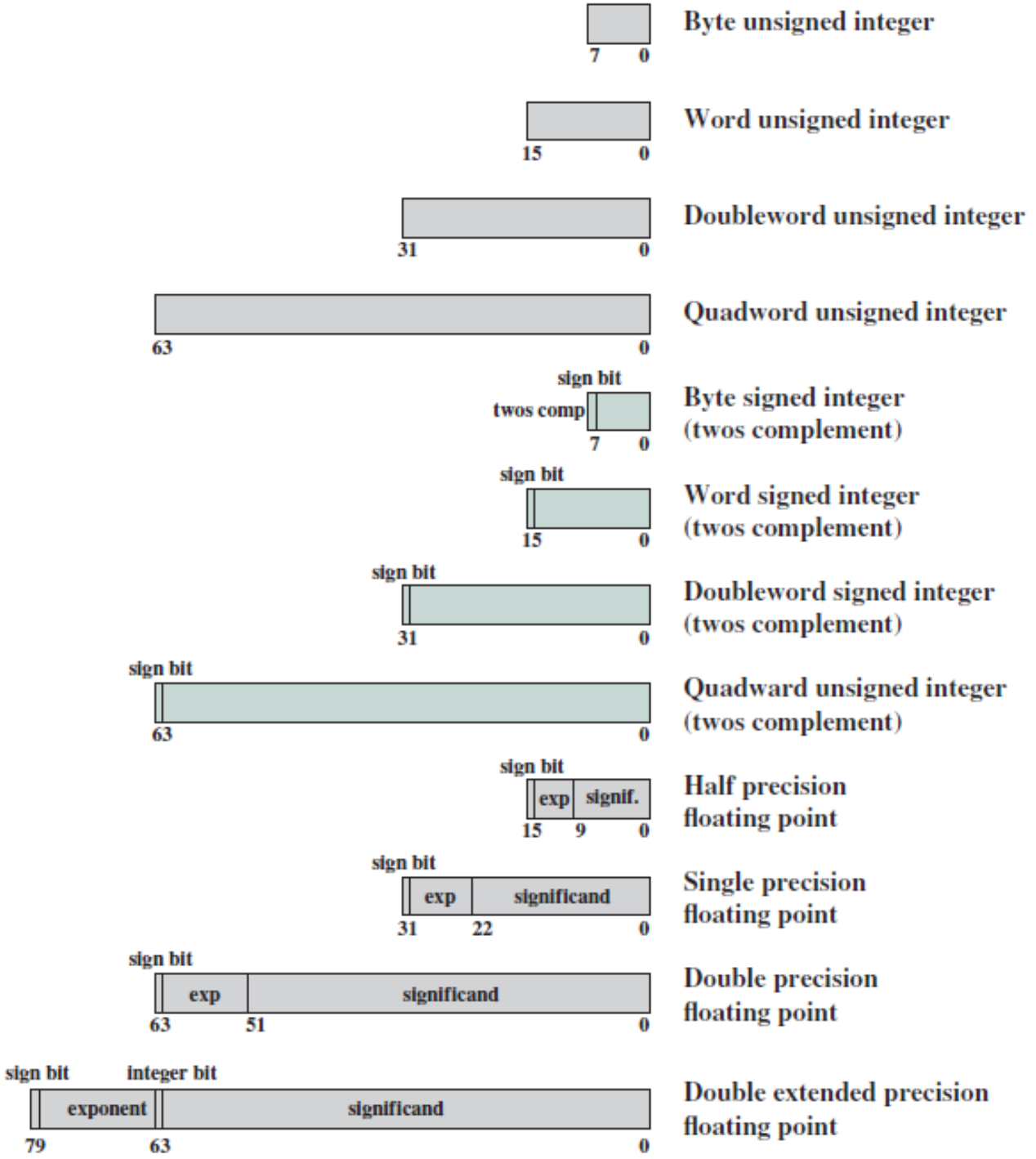
TYPES OF OPERANDS

- General categories of data are
  - Addresses
  - Numbers
  - Characters
  - Logical data

x86 Data Typ

Data Type	Description
General	Byte, word (16 bits), doubleword (32 bits), quadword (64 bits), and double quadword (128 bits) locations with arbitrary binary contents.
Integer	A signed binary value contained in a byte, word, or doubleword, using twos complement representation.
Ordinal	An unsigned integer contained in a byte, word, or doubleword.
Unpacked binary coded decimal (BCD)	A representation of a BCD digit in the range 0 through 9, with one digit in each byte.
Packed BCD	Packed byte representation of two BCD digits; value in the range 0 to 99.
Near pointer	A 16-bit, 32-bit, or 64-bit effective address that represents the offset within a segment. Used for all pointers in a nonsegmented memory and for references within a segment in a segmented memory.
Far pointer	A logical address consisting of a 16-bit segment selector and an offset of 16, 32, or 64 bits. Far pointers are used for memory references in a segmented memory model where the identity of a segment being accessed must be specified explicitly.
Bit field	A contiguous sequence of bits in which the position of each bit is considered as an independent unit. A bit string can begin at any bit position of any byte and can contain up to 32 bits.
Bit string	A contiguous sequence of bits, containing from zero to $2^{23} - 1$ bits.
Byte string	A contiguous sequence of bytes, words, or doublewords, containing from zero to $2^{23} - 1$ bytes.
Floating point	See Figure 12.4.
Packed SIMD (single instruction, multiple data)	Packed 64-bit and 128-bit data types.

x86 Numeric Data Typ



TYPES OF OPERATIONS

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

TYPES OF 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

TYPES OF OPERATIONS

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
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

TYPES OF OPERATIONS

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)

TYPES OF OPERATIONS

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

- The data transfer instruction must specify
  - The location of the source and destination operands. Each location could be memory, a register, or the top of the stack.
  - The length of data to be transferred
  - The mode of addressing for each operand must be specified.
- If both source and destination are registers, then the processor simply causes data to be transferred from one register to another; this is an operation internal to the processor.
- If one or both operands are in memory, then the processor must perform following actions:
  1. Calculate the memory address, based on the address mode.
  2. If the address refers to virtual memory, translate from virtual to real memory address.
  3. Determine whether the addressed item is in cache.
  4. If not, issue a command to the memory module.

Data Transfe

Table 12.5 Examples of IBM EAS/390 Data Transfer Operations

Operation Mnemonic	Name	Number of Bits Transferred	Description
L	Load	32	Transfer from memory to register
LH	Load Halfword	16	Transfer from memory to register
LR	Load	32	Transfer from register to register
LER	Load (short)	32	Transfer from floating-point register to floating-point register
LE	Load (short)	32	Transfer from memory to floating-point register
LDR	Load (long)	64	Transfer from floating-point register to floating-point register
LD	Load (long)	64	Transfer from memory to floating-point register
ST	Store	32	Transfer from register to memory
STH	Store Halfword	16	Transfer from register to memory
STC	Store Character	8	Transfer from register to memory
STE	Store (short)	32	Transfer from floating-point register to memory
STD	Store (long)	64	Transfer from floating-point register to memory

Arithmetic

- Most machines provide the basic arithmetic operations of add, subtract, multiply, and divide. These are invariably provided for signed integer numbers.
- The execution of an arithmetic instruction may involve data transfer operations to position operands for input to the ALU, and to deliver the output of the ALU

Logical

- Most machines also provide a variety of operations for manipulating individual bits of a word or other addressable units, often referred to as "bit twiddling." They are based upon Boolean operations.

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

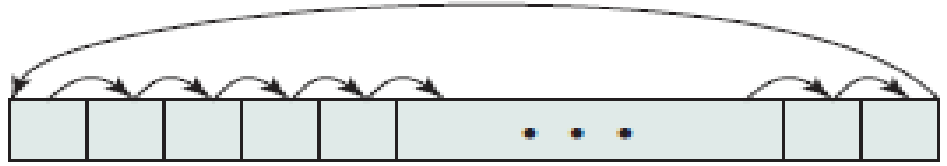
Logical

(a) Logical right shift

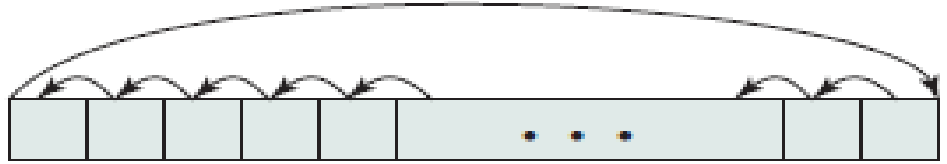
(b) Logical left shift

(c) Arithmetic right shift

(d) Arithmetic left shift



(e) Right rotate



(f) Left rotate

Conversion

- Conversion instructions are those that change the format or operate on the format of data. An example is converting from decimal to binary.

Input/Output

- There are a variety of approaches taken, including isolated programmed I/O, memory-mapped programmed I/O, DMA, and the 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.

Transfer of Control

- The next instruction to be performed is the one that immediately follows, in memory, the current instruction.
- A significant fraction of the instructions in any program have as their function changing the sequence of instruction execution. For these instructions, the operation performed by the processor is to update the program counter to contain the address of some instruction in memory.
- The most common transfer-of-control operations are branch, skip, and procedure call.
- Branch can be either forward (an instruction with a higher address) or backward