电子科技大学
格拉斯哥学院
Glasgow College, UESTC

UESTC4019: Real-Time Computer Systems and Architecture

Lecture 13

Instruction Sets – Characteristics and Function (Part-2)

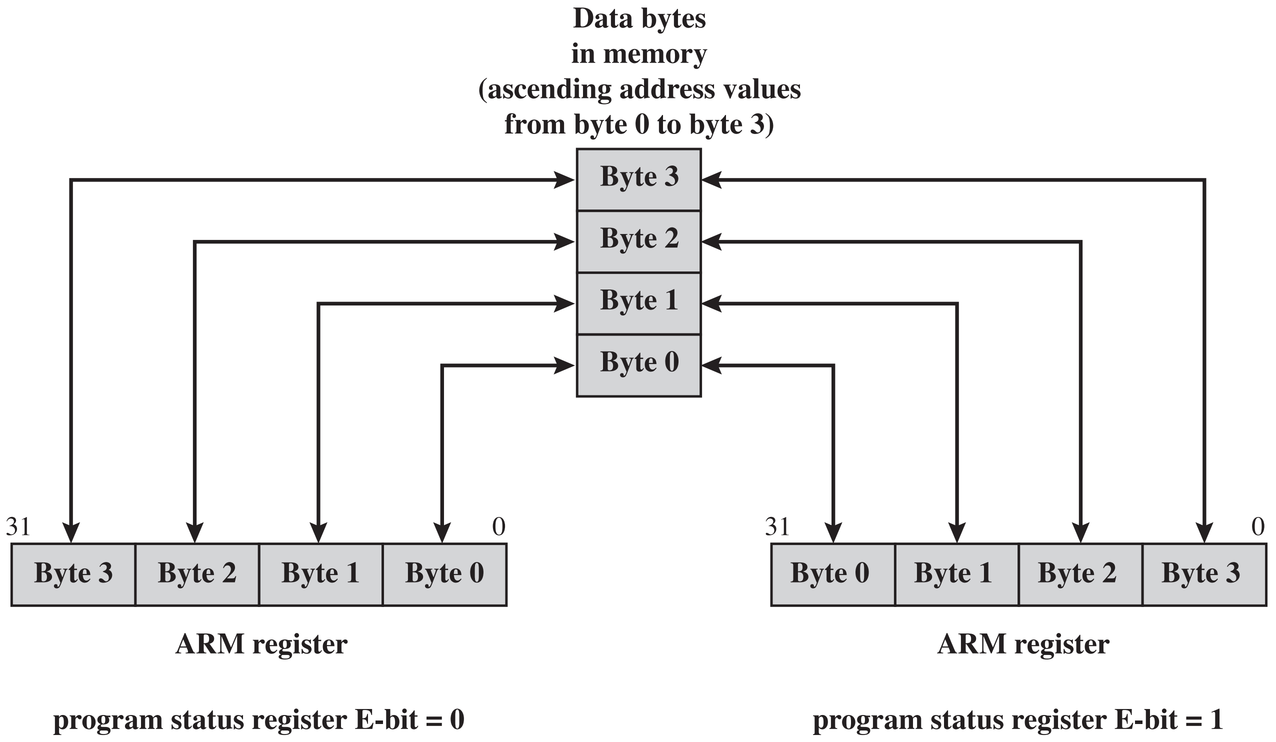
ARM Data Types (1 of 2)

- ARM processors support data types of:
  - 8 (byte)
  - 16 (half word)
  - 32 (word) bits in length
- Alignment checking
  - When the appropriate control bit is set, a data abort signal indicates an alignment fault for attempting unaligned access

ARM Data Types (2 of 2)

- Unaligned access
  - When this option is enabled, the processor uses one or more memory accesses to generate the required transfer of adjacent bytes transparently to the programmer
- For all three data types an unsigned interpretation is supported in which the value represents an unsigned, nonnegative integer
- All three data types can also be used for twos complement signed integers

ARM Endian Support Word Load/Store with E-bit



Common Instruction Set Operations (1 of 5)

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

Common Instruction Set Operations (2 of 5)

Type	Operation Name	Description
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

Common Instruction Set Operations (3 of 5)

Type	Operation Name	Description
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

Common Instruction Set Operations (4 of 5)

Type	Operation Name	Description
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

Common Instruction Set Operations (5 of 5)

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

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

Examples of IBM EAS/390 Data Transfer Operations (1 of 2)

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

Examples of IBM EAS/390 Data Transfer Operations (2 of 2)

Operation Mnemonic	**Name**	**Number of Bits Transferred**	**Description**
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 (1 of 2)

- Most machines provide the basic arithmetic operations of add, subtract, multiply, and divide
- These are provided for signed integer (fixed-point) numbers
- Often they are also provided for floating-point and packed decimal numbers
- Other possible operations include a variety of single-operand instructions:
  - Absolute
    - Take the absolute value of the operand

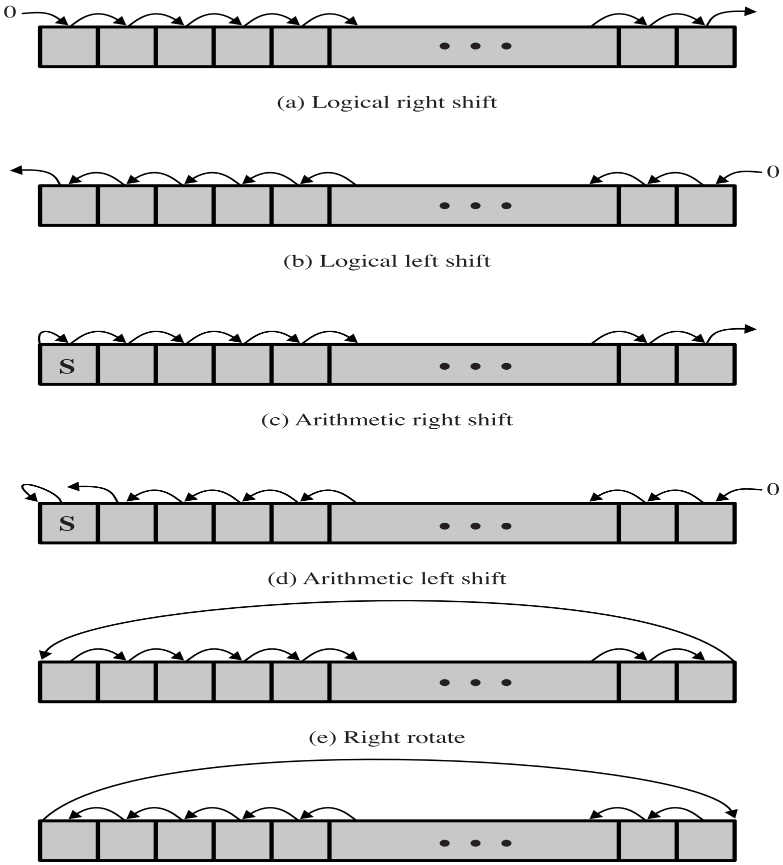
Arithmetic (2 of 2)

- Negate
  - Negate the operand
- Increment
  - Add 1 to the operand
- Decrement
  - Subtract 1 from the operand

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

Shift and Rotate Operations



(a) Logical right shift

(b) Logical left shift

(c) Arithmetic right shift

(d) Arithmetic left shift

(e) Right rotate

(f) Left rotate

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 E AS/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

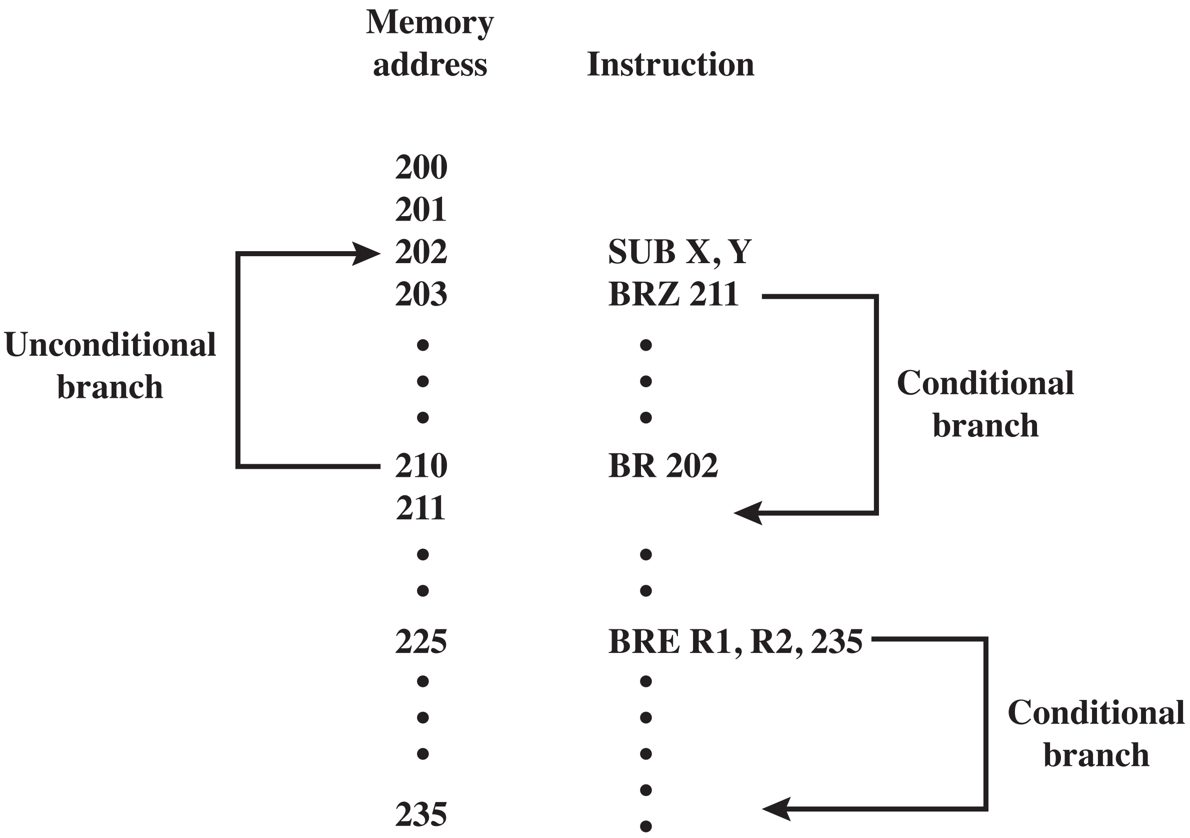
System Control

- 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

Branch Instructions



Memory address	Instruction
200	
201	
202	SUB X, Y
203	BRZ 211
•	•
•	•
•	•
210	BR 202
211	
•	•
•	•
225	BRE R1, R2, 235
•	•
•	•
•	•
	•
235	•

Unconditional branch

Conditional branch

Conditional branch

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

Procedure Call Instructions (1 of 2)

- 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

Procedure Call Instructions (2 of 2)

- 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

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

ARM Conditions for Conditional Instruction Execution (1 of 2)

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

ARM Conditions for Conditional Instruction Execution (2 of 2)

Code	Symbol	Condition Tested	Comment
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