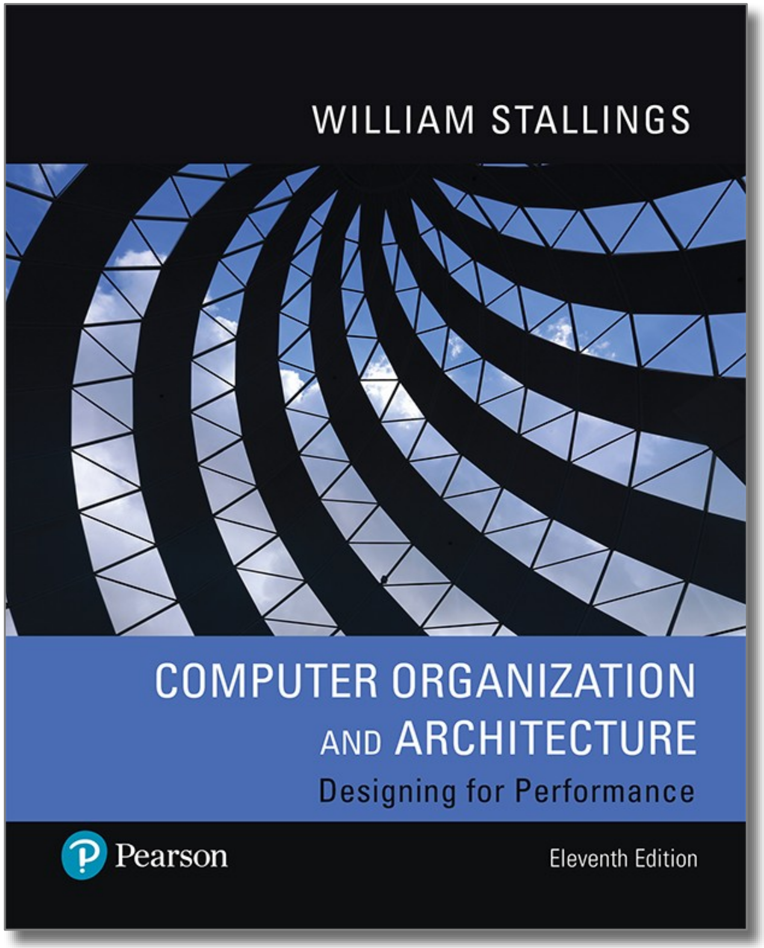
Computer Organization and Architecture

Designing for Performance

11th Edition



Chapter 14

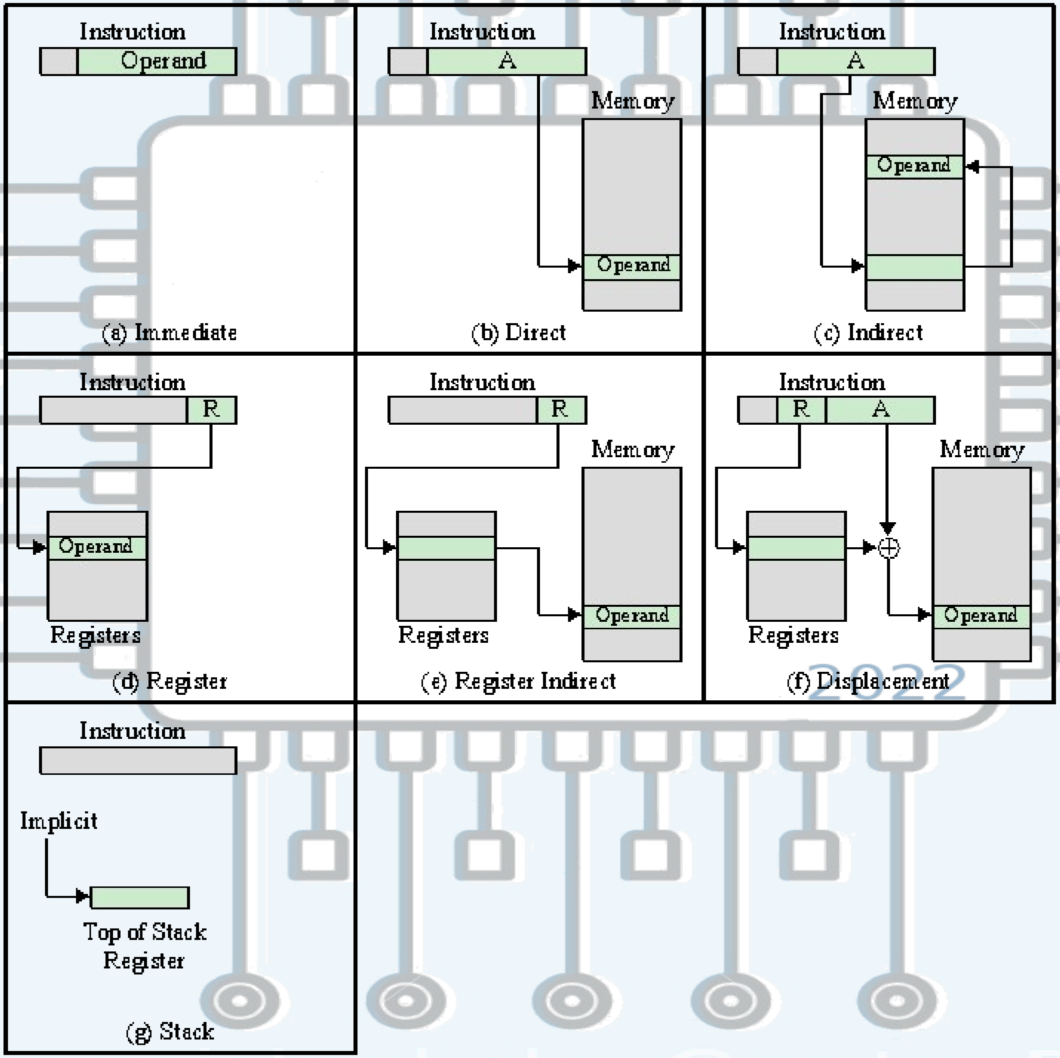
Instruction Sets: Addressing Modes and Formats

Addressing Modes

- Immediate
- Direct
- Indirect
- Register
- Register indirect
- Displacement
- Stack

ECCI
Universidad de Costa Rica
Fco. Arroyo

Figure 14.1
Addressing Modes



Universidad de Costa Rica
ECCI
2022
Fco. Arroyo

Table 14.1
Basic Addressing Modes

Mode	Algorithm	Principal Advantage	Principal Disadvantage
Immediate	Operand = A	No memory reference	Limited operand magnitude
Direct	EA = A	Simple	Limited address space
Indirect	EA = (A)	Large address space	Multiple memory references
Register	EA = R	No memory reference	Limited address space
Register indirect	EA = (R)	Large address space	Extra memory reference
Displacement	EA = A + (R)	Flexibility	Complexity
Stack	EA = top of stack	No memory reference	Limited applicability

ECCI
Universidad de Costa Rica
Fco. Arroyo

Immediate Addressing

- Simplest form of addressing
- Operand = A
- This mode can be used to define and use constants or set initial values of variables
  - Typically the number will be stored in twos complement form
  - The leftmost bit of the operand field is used as a sign bit
- Advantage:
  - No memory reference other than the instruction fetch is required to obtain the operand, thus saving one memory or cache cycle in the instruction cycle
- Disadvantage:
  - The size of the number is restricted to the size of the address field, which, in most instruction sets, is small compared with the word length

2022
ECCI
Universidad de Costa Rica
Fco. Arroyo

Direct Addressing

Address field contains the effective address of the operand

Effective address (EA) = address field (A)

Was common in earlier generations of computers

Requires only one memory reference and no special calculation

Limitation is that it provides only a limited address space

Universidad de Costa Rica

Fco. Arroyo

Indirect Addressing

- Reference to the address of a word in memory which contains a full-length address of the operand
- EA = (A)
  - Parentheses are to be interpreted as meaning *contents of*
- Advantage:
  - For a word length of *N* an address space of 2^N is now available
- Disadvantage:
  - Instruction execution requires two memory references to fetch the operand
    - One to get its address and a second to get its value
- A rarely used variant of indirect addressing is multilevel or cascaded indirect addressing
  - EA = (. . . (A) . . .)
  - Disadvantage is that three or more memory references could be required to fetch an operand

2022
ECCI
Universidad de Costa Rica
Fco. Arroyo

Register Addressing

Address field refers to a register rather than a main memory address

EA = R

Advantages:

- Only a small address field is needed in the instruction
- No time-consuming memory references are required

Disadvantage:

- The address space is very limited

ECCI
Universidad de Costa Rica
Fco. Arroyo

Register Indirect Addressing

- Analogous to indirect addressing
  - The only difference is whether the address field refers to a memory location or a register
- EA = (R)
- Address space limitation of the address field is overcome by having that field refer to a word-length location containing an address
- Uses one less memory reference than indirect addressing

2022
ECCI
Universidad de Costa Rica
Fco. Arroyo

Displacement Addressing

- Combines the capabilities of direct addressing and register indirect addressing
- EA = A + (R)
- Requires that the instruction have two address fields, at least one of which is explicit
  - The value contained in one address field (value = A) is used directly
  - The other address field refers to a register whose contents are added to A to produce the effective address
- Most common uses:
  - Relative addressing
  - Base-register addressing
  - Indexing

2022
ECCI
Universidad de Costa Rica
Fco. Arroyo

Relative Addressing

The implicitly referenced register is the program counter (PC)

- The next instruction address is added to the address field to produce the EA
- Typically the address field is treated as a twos complement number for this operation
- Thus the effective address is a displacement relative to the address of

Exploits the concept of locality

Saves address bits in the instruction if most memory references are relatively near to the instruction being executed

Fco. Arroyo

Universidad de Costa Rica

Base-Register Addressing

- The referenced register contains a main memory address and the address field contains a displacement from that address
- The register reference may be explicit or implicit
- Exploits the locality of memory references
- Convenient means of implementing segmentation
- In some implementations a single segment base register is employed and is used implicitly
- In others the programmer may choose a register to hold the base address of a segment and the instruction must reference it explicitly

2022 ECCI Universidad de Costa Rica

Indexing

- The address field references a main memory address and the referenced register contains a positive displacement from that address
- The method of calculating the EA is the same as for base-register addressing
- An important use is to provide an efficient mechanism for performing iterative operations
- Autoindexing
  - Automatically increment or decrement the index register after each reference to it
  - EA = A + (R)
  - (R) □(R) + 1
- Postindexing
  - Indexing is performed after the indirection
  - EA = (A) + (R)
- Preindexing
  - Indexing is performed before the indirection
  - EA = (A + (R))

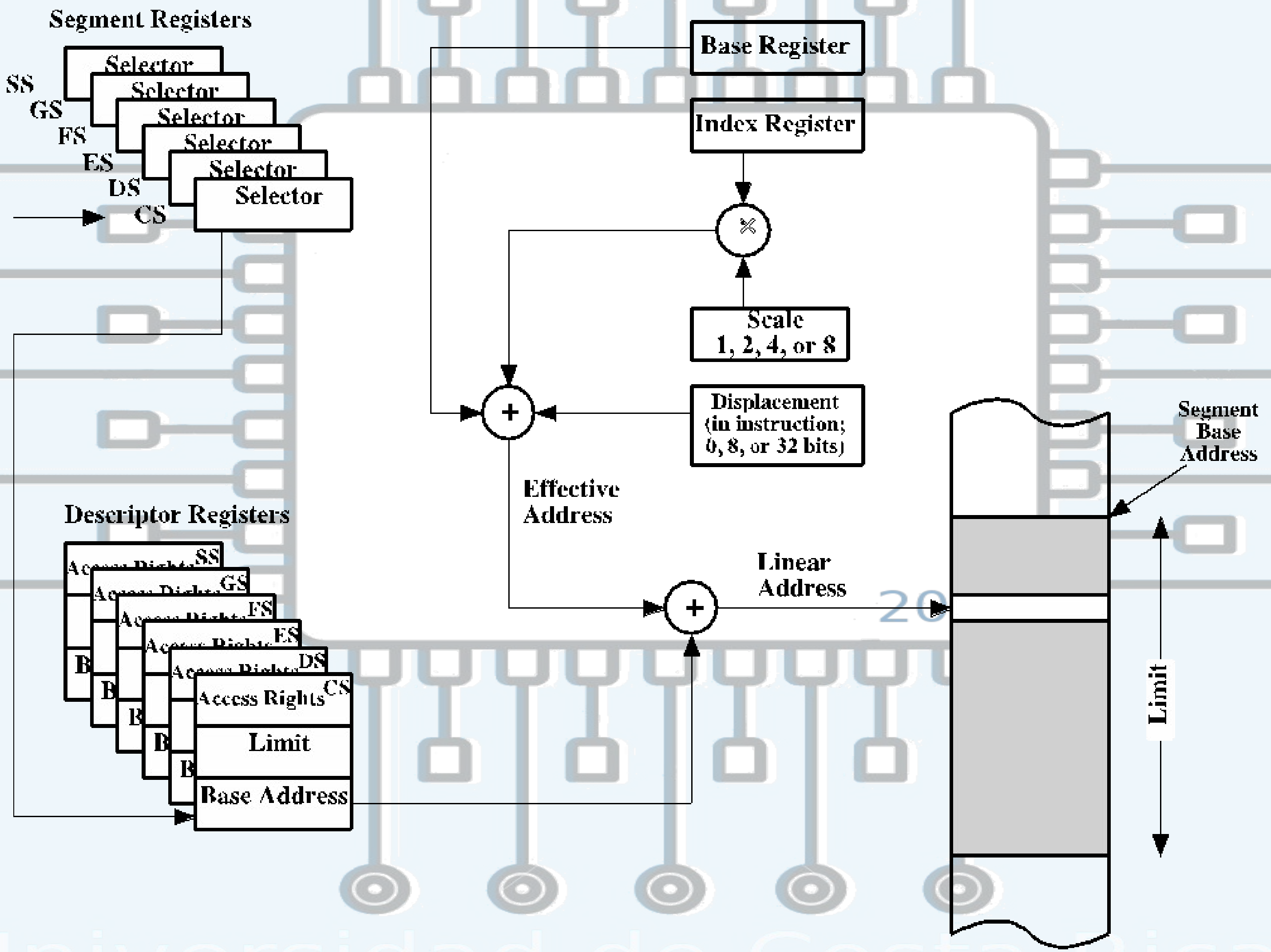
2022
ECCI
Universidad de Costa Rica
Fco. Arroyo

Stack Addressing

- A stack is a linear array of locations
  - Sometimes referred to as a *pushdown list* or *last-in-first-out queue*
- A stack is a reserved block of locations
  - Items are appended to the top of the stack so that the block is partially filled
- Associated with the stack is a pointer whose value is the address of the top of the stack
  - The stack pointer is maintained in a register
  - Thus references to stack locations in memory are in fact register indirect addresses
- Is a form of implied addressing
- The machine instructions need not include a memory reference but implicitly operate on the top of the stack

2022
ECCI
Universidad de Costa Rica
Fco. Arroyo

Figure 14.2
x86 Addressing Mode Calculation



Universidad de Costa Rica
ECCI
Fco. Arroyo

Table 14.2
x86 Addressing Modes

Mode	Algorithm
Immediate	Operand = A
Register Operand	LA = R
Displacement	LA = (SR) + A
Base	LA = (SR) + (B)
Base with Displacement	LA = (SR) + (B) + A
Scaled Index with Displacement	LA = (SR) + (I) × S + A
Base with Index and Displacement	LA = (SR) + (B) + (I) + A
Base with Scaled Index and Displacement	LA = (SR) + (I) × S + (B) + A
Relative	LA = (PC) + A

LA = linear address
(X) = contents of X
SR = segment register
PC = program counter
A = contents of an address field in the instruction

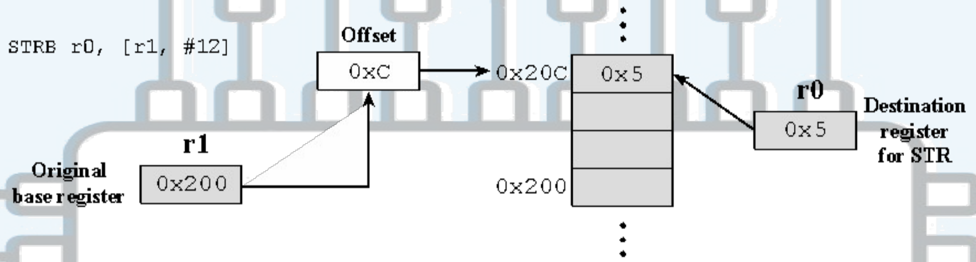
R = register
B = base register
I = index register
S = scaling factor

ECCI

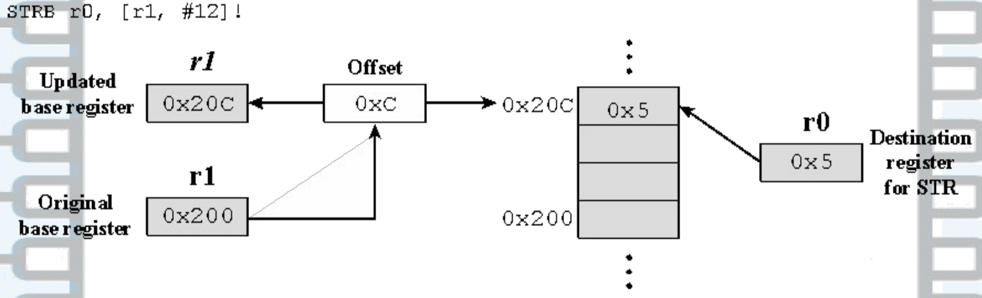
Universidad de Costa Rica

Fco. Arroyo

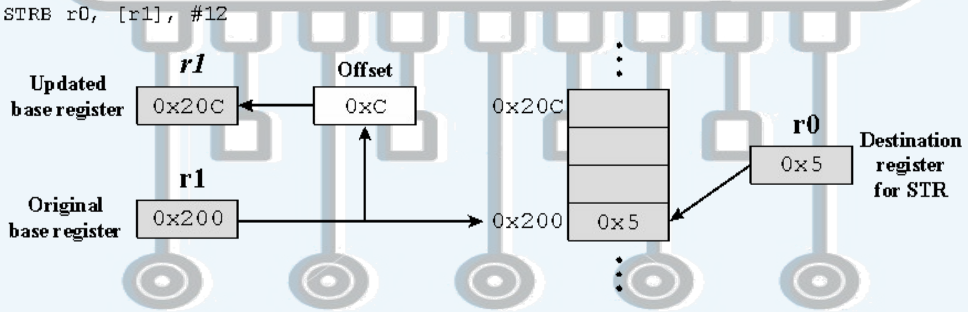
Figure 14.3
ARM Indexing Methods



(a) Offset



(b) Preindex



(c) Postindex

2022
ECCI
Universidad de Costa Rica
Fco. Arroyo

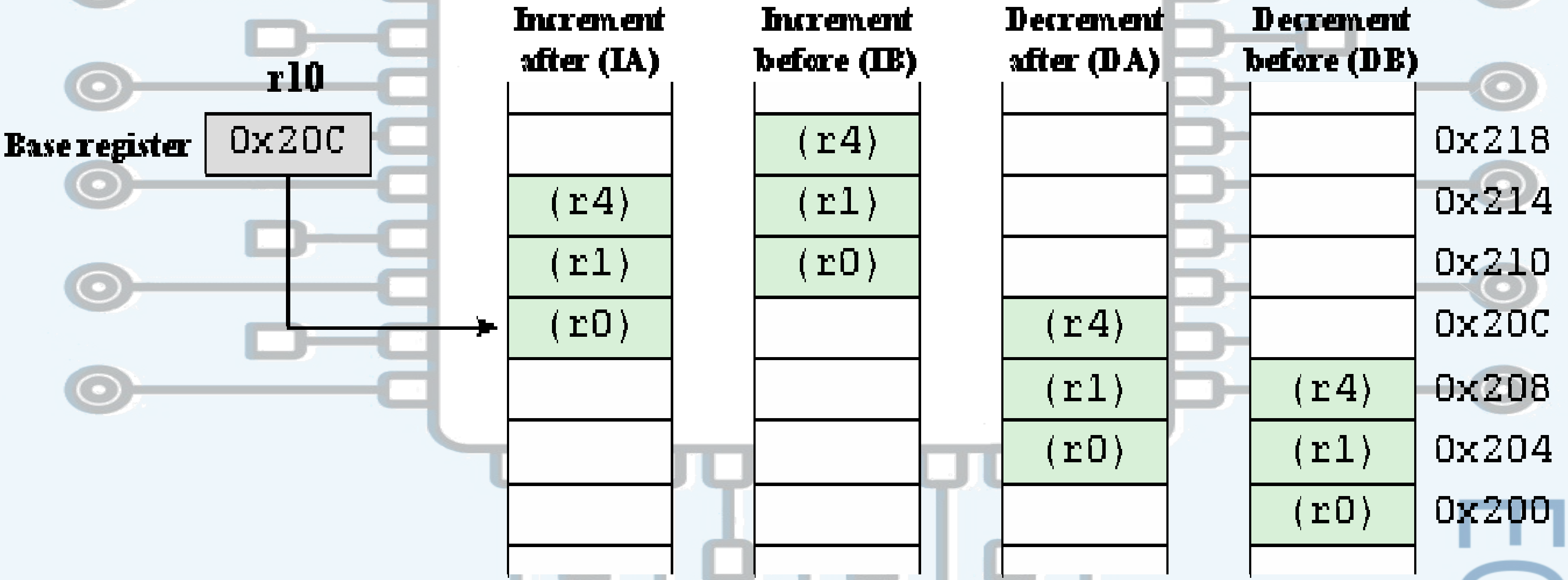
ARM Data Processing Instruction Addressing and Branch Instructions

- Data processing instructions
  - Use either register addressing or a mixture of register and immediate addressing
  - For register addressing the value in one of the register operands may be scaled using one of the five shift operators
- Branch instructions
  - The only form of addressing for branch instructions is immediate
  - Instruction contains 24 bit value
    - Shifted 2 bits left so that the address is on a word boundary
    - Effective range ± 32MB from from the program counter

2022 ECCI Universidad de Costa Rica Fco. Arroyo

Figure 14.4
ARM Load/Store Multiple Addressing

```
LDMxx r10, {r0, r1, r4}
STMxx r10, {r0, r1, r4}
```



	Increment after (IA)	Increment before (IB)	Decrement after (DA)	Decrement before (DB)
0x218		(r4)		
0x214	(r4)	(r1)		
0x210	(r1)	(r0)		
0x20C	(r0)		(r4)	
0x208			(r1)	(r4)
0x204			(r0)	(r1)
0x200				(r0)

Universidad de Costa Rica
ECCI
Fco. Arroyo

Instruction Formats

Define the layout of the bits of an instruction, in terms of its constituent fields

Must include an opcode and, implicitly or explicitly, indicate the addressing mode for each operand

For most instruction sets more than one instruction format is used

ECCI
Universidad de Costa Rica
Fco. Arroyo

Instruction Length

- Most basic design issue
- Affects, and is affected by:
  - Memory size
  - Memory organization
  - Bus structure
  - Processor complexity
  - Processor speed
- Should be equal to the memory-transfer length or one should be a multiple of the other
- Should be a multiple of the character length, which is usually 8 bits, and of the length of fixed-point numbers

2022 ECCI Universidad de Costa Rica Fco. Arroyo

Allocation of Bits

- Number of addressing modes
- Number of operands
- Register versus memory
- Number of register sets
- Address range
- Address granularity

Universidad de Costa Rica

Fco. Arroyo

Figure 14.5
PDP-8 Instruction Formats

Memory Reference Instructions

Opcode			D/I	Z/C	Displacement						
0		2	3	4	5						11

Input/Output Instructions

1	1	0	Device						Opcode		
0		2	3					8	9		11

Register Reference Instructions

Group 1 Microinstructions

1	1	1	0	CLA	CLL	CMA	CML	RAR	RAL	BSW	IAC
0	1	2	3	4	5	6	7	8	9	10	11

Group 2 Microinstructions

1	1	1	1	CLA	SMA	SZA	SNL	RSS	OSR	HLT	0
0	1	2	3	4	5	6	7	8	9	10	11

Group 3 Microinstructions

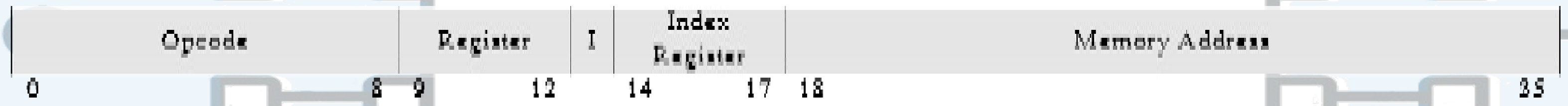
1	1	1	1	CLA	MQA	0	MQL	0	0	0	1
0	1	2	3	4	5	6	7	8	9	10	11

D/I = Direct/Indirect address
Z/C = Page 0 or Current page
CLA = Clear Accumulator
CLL = Clear Link
CMA = CoMplement Accumulator
CML = CoMplement Link
RAR = Rotate Accumultator Right
RAL = Rotate Accumulator Left
BSW = Byte SWap

IAC = Increment ACcumulator
SMA = Skip on Minus Accumulator
SZA = Skip on Zero Accumulator
SNL = Skip on Nonzero Link
RSS = Reverse Skip Sense
OSR = Or with Switch Register
HLT = HaLT
MQA = Multiplier Quotient into Accumulator
MQL = Multiplier Quotient Load

2022
ECCI
Universidad de Costa Rica
Fco. Arroyo

Figure 14.6
PDP-10 Instruction Format



I = indirect bit

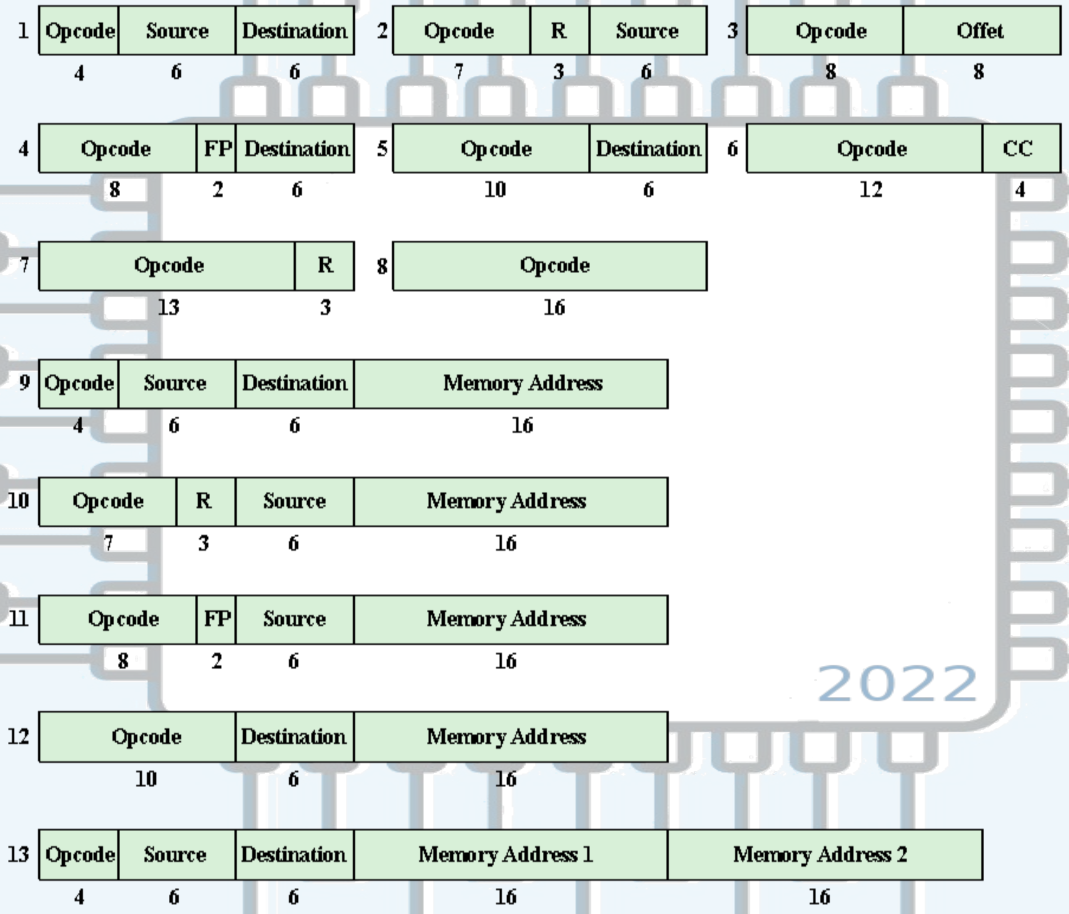
2022
ECCI
Universidad de Costa Rica
Fco. Arroyo

Variable-Length Instructions

- Variations can be provided efficiently and compactly
- Increases the complexity of the processor
- Does not remove the desirability of making all of the instruction lengths integrally related to word length
  - Because the processor does not know the length of the next instruction to be fetched a typical strategy is to fetch a number of bytes or words equal to at least the longest possible instruction
  - Sometimes multiple instructions are fetched

2022 ECCI Universidad de Costa Rica Fco. Arroyo

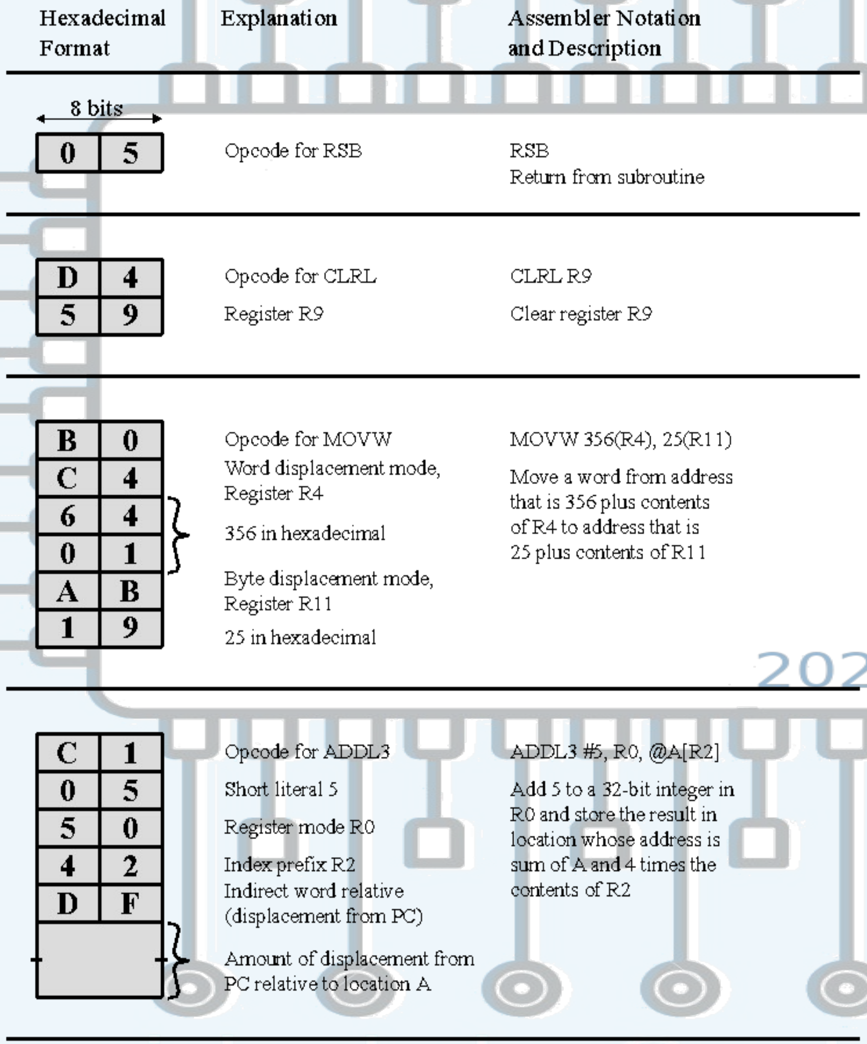
Figure 14.7
Instruction Formats for the PDP-11



Numbers below fields indicate bit length
Source and Destination each contain a 3-bit addressing mode field and a 3-bit register number
FP indicates one of four floating-point registers
R indicates one of the general-purpose registers
CC is the condition code field

2022
ECCI
Universidad de Costa Rica
Fco. Arroyo

Figure 14.8
Example of VAX Instructions



8 bits

Hexadecimal Format		Explanation	Assembler Notation and Description
0	5	Opcode for RSB	RSB Return from subroutine
D	4	Opcode for CLRL	CLRL R9
5	9	Register R9	Clear register R9
B	0	Opcode for MOVW	MOVW 356(R4), 25(R11)
C	4	Word displacement mode, Register R4	Move a word from address that is 356 plus contents of R4 to address that is 25 plus contents of R11
6	4	356 in hexadecimal	
0	1		
A	B	Byte displacement mode, Register R11	
1	9	25 in hexadecimal	
C	1	Opcode for ADDL3	ADDL3 #5, R0, @A[R2]
0	5	Short literal 5	Add 5 to a 32-bit integer in R0 and store the result in location whose address is sum of A and 4 times the contents of R2
5	0	Register mode R0	
4	2	Index prefix R2	
D	F	Indirect word relative (displacement from PC)	
		Amount of displacement from PC relative to location A	

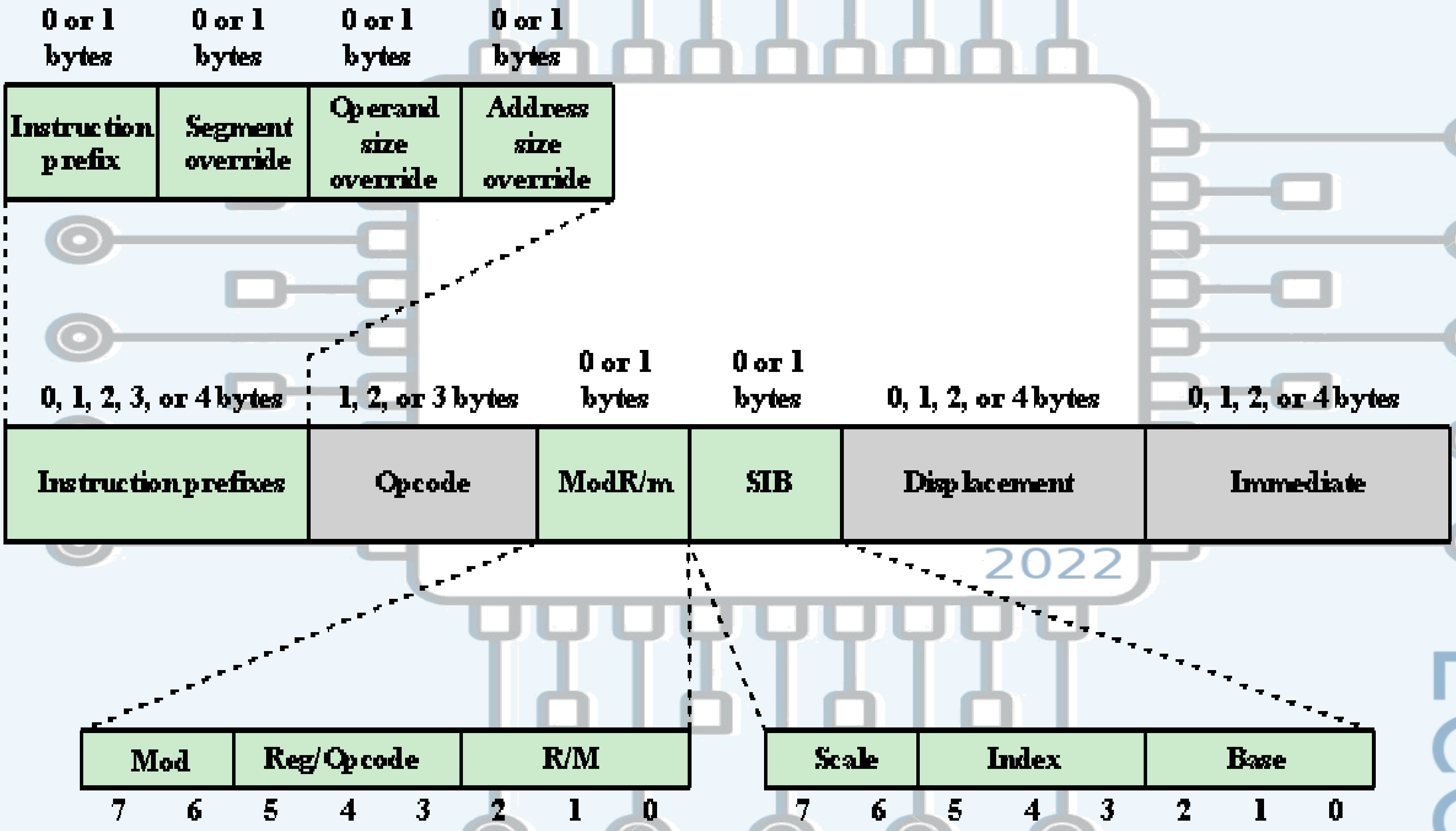
2022

ECCI

Universidad de Costa Rica

Fco. Arroyo

Figure 14.9
x86 Instruction Format



Instruction prefix	Segment override	Operand size override	Address size override
0 or 1 bytes	0 or 1 bytes	0 or 1 bytes	0 or 1 bytes

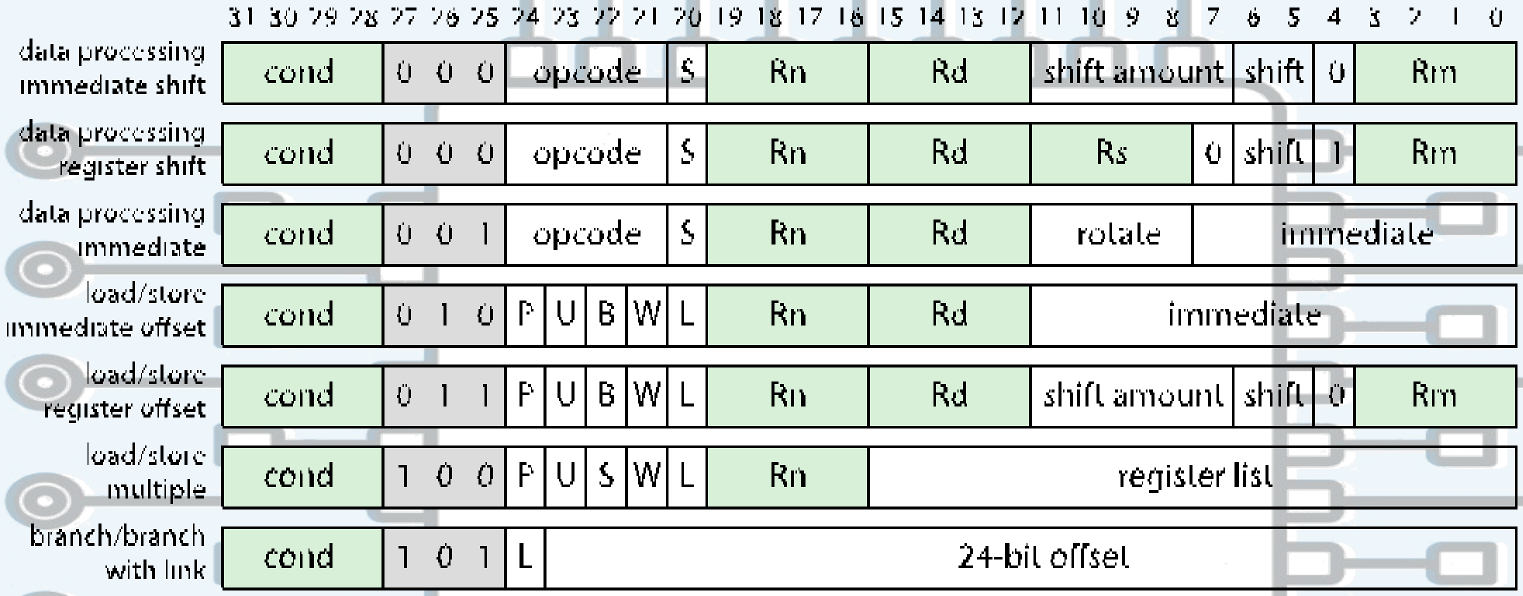
Instruction prefixes	Opcode	ModR/m	SIB	Displacement	Immediate
0, 1, 2, 3, or 4 bytes	1, 2, or 3 bytes	0 or 1 bytes	0 or 1 bytes	0, 1, 2, or 4 bytes	0, 1, 2, or 4 bytes

Mod	Reg/Opcode	R/M
7 6	5 4 3	2 1 0

Scale	Index	Base
7 6	5 4 3	2 1 0

2022
ECCI
Universidad de Costa Rica
Fco. Arroyo

Figure 14.10
ARM Instruction Formats



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
data processing immediate shift	cond				0	0	0	opcode				S	Rn				Rd				shift amount					shift		0	Rm			
data processing register shift	cond				0	0	0	opcode				S	Rn				Rd				Rs				0	shift		1	Rm			
data processing immediate	cond				0	0	1	opcode				S	Rn				Rd				rotate				immediate							
load/store immediate offset	cond				0	1	0	P	U	B	W	L	Rn				Rd				immediate											
load/store register offset	cond				0	1	1	P	U	B	W	L	Rn				Rd				shift amount					shift		0	Rm			
load/store multiple	cond				1	0	0	P	U	S	W	L	Rn				register list															
branch/branch with link	cond				1	0	1	L	24-bit offset																							

S – For data processing instructions, signifies that the instruction updates the condition codes

S – For load/store multiple instructions, signifies whether instruction execution is restricted to supervisor mode

P, U, W = bits that distinguish among different types of addressing_mode

B – Distinguishes between an unsigned byte (B==1) and a word (B==0) access

L – For load/store instructions, distinguishes between a Load (L==1) and a Store (L==0)

L = For branch instructions, determines whether a return address is stored in the link register

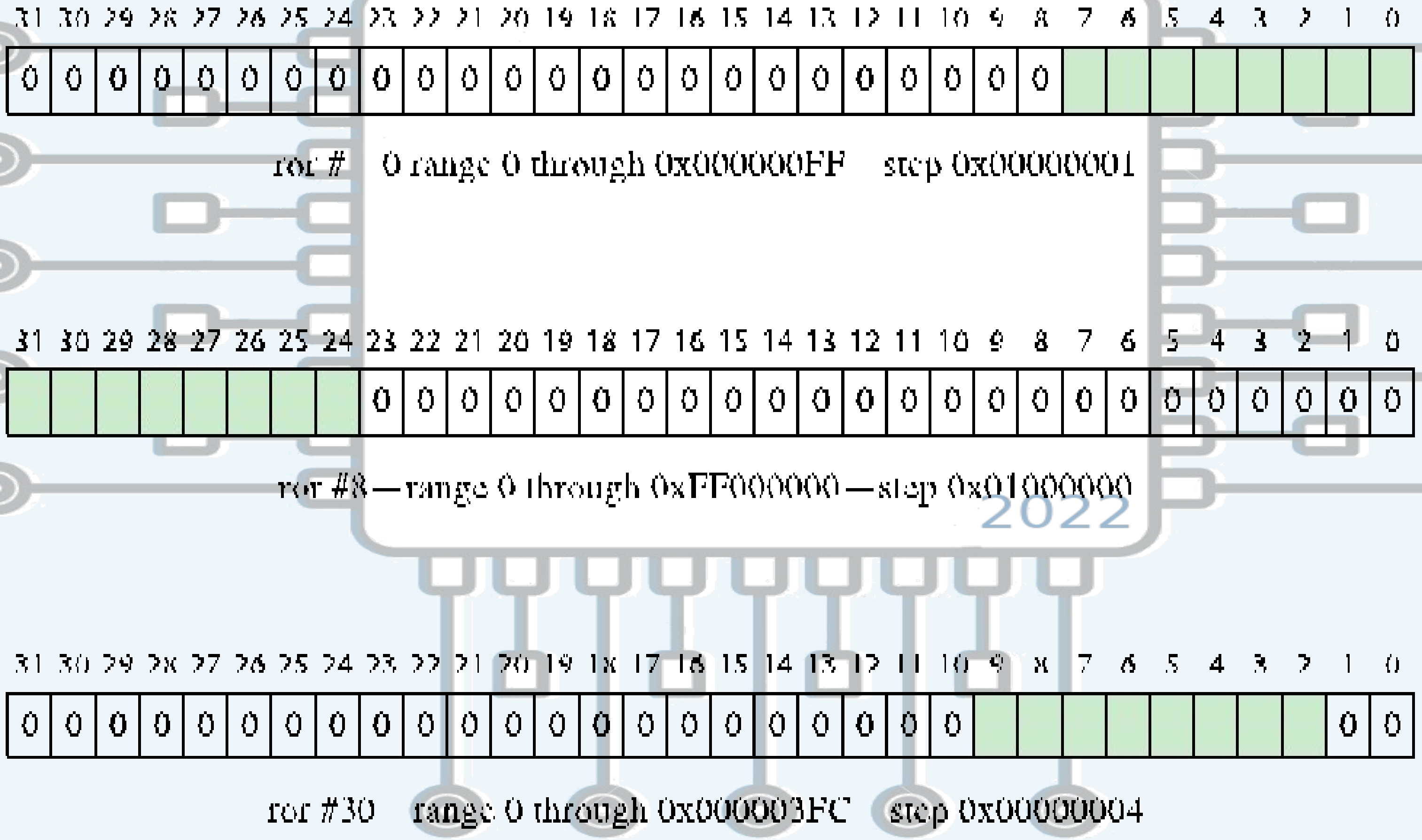
2022

ECCI

Universidad de Costa Rica

Fco. Arroyo

Figure 14.11
Examples of Use of ARM Immediate Constants



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩

ror #0 range 0 through 0x000000FF step 0x00000001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ror #8—range 0 through 0xFF000000—step 0x01000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	0	0

ror #30 range 0 through 0x000003FC step 0x00000004

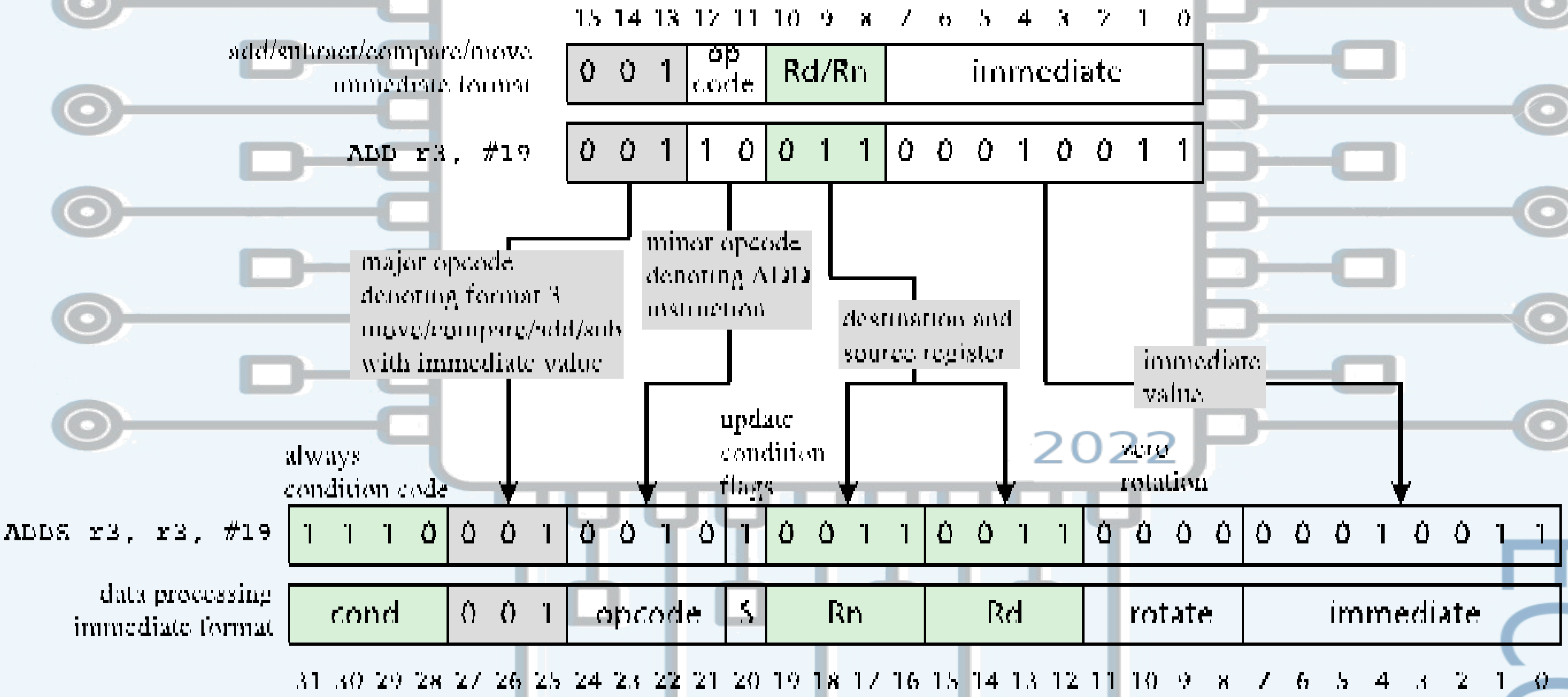
2022

ECCI

Universidad de Costa Rica

Fco. Arroyo

Figure 14.12
Expanding a Thumb ADD Instruction into its ARM Equivalent



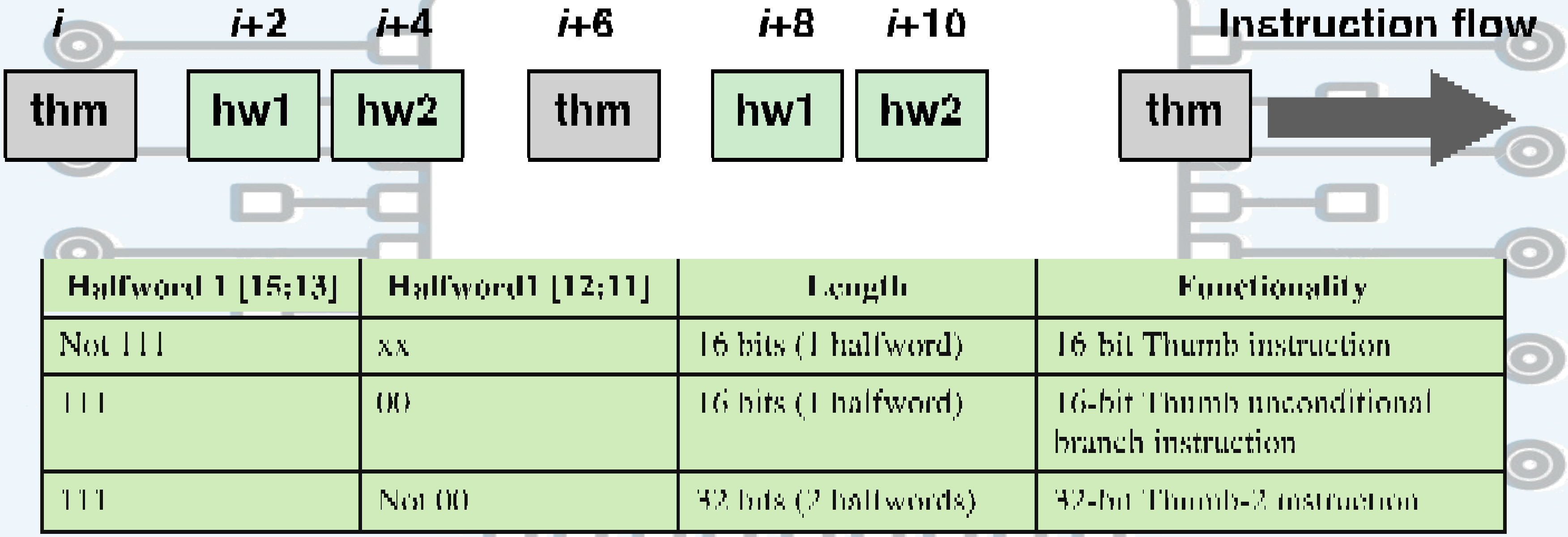
2022
ECCI
Universidad de Costa Rica
Fco. Arroyo

Thumb-2 Instruction Set

- The only instruction set available on the Cortex-M microcontroller products
- Is a major enhancement to the Thumb instruction set architecture (ISA)
  - Introduces 32-bit instructions that can be intermixed freely with the older 16-bit Thumb instructions
  - Most 32-bit Thumb instructions are unconditional, whereas almost all ARM instructions can be conditional
  - Introduces a new If-Then (IT) instruction that delivers much of the functionality of the condition field in ARM instructions
- Delivers overall code density comparable with Thumb, together with the performance levels associated with the ARM ISA
- Before Thumb-2 developers had to choose between Thumb for size and ARM for performance

Universidad de Costa Rica ECCI 2022 Fco. Arroyo

Figure 14.13
Thumb-2 Encoding



Halfword 1 [15:13]	Halfword1 [12:11]	Length	Functionality
Not 111	xx	16 bits (1 halfword)	16 bit Thumb instruction
111	00	16 bits (1 halfword)	16-bit Thumb unconditional branch instruction
111	Not 00	32 bits (2 halfwords)	32-bit Thumb-2 instruction

ECCI
Universidad de Costa Rica
Fco. Arroyo

Summary

•Instruction Sets: Addressing Modes and Formats

Chapter 14

- Addressing modes
  - Immediate addressing
  - Direct addressing
  - Indirect addressing
  - Register addressing
  - Register indirect addressing
  - Displacement addressing
  - Stack addressing

- x86 addressing modes
- ARM addressing modes
- Instruction formats
  - Instruction length
  - Allocation of bits
  - Variable-length instructions
- X86 instruction formats
- ARM instruction formats

2022
ECCI
Universidad de Costa Rica
Fco. Arroyo