



William Stallings Computer Organization and Architecture 10th Edition

+ Chapter 13

Instruction Sets: Addressing Modes and Formats



Immediate

Direct

Indirect

Register

Register indirect

Displacement

 $x \in X$

Stack



Figure 13.1 Addressing Modes

Table 13.1 Basic Addressing Modes

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

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

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

(D

Limitation is that it provides only a limited address space

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

Register Addressing

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

 $\mathbf{E}\mathbf{A} = \mathbf{R}$

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

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

Displacement Addressing

- Combines the capabilities of direct addressing and register indirect addressing
- $\blacksquare EA = A + (R)$
- 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

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 the instruction

Exploits the concept of locality

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

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

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))

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





Table 13.2 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	$LA = (SR) + (I) \times S + A$
Base with Index and Displacement	LA = (SR) + (B) + (I) + A
Base with Scaled Index and Displacement	$LA = (SR) + (I) \times S + (B) + A$
Relative	LA = (PC) + A
LA = linear address (X) = contents of X SR = segment register	

program counter PC

contents of an address field in the instruction Α =

R register =

base register В = Ι

index register =

scaling factor S =

STRB r0, [r1, #12]





STRB r0, [r1, #12]!



(b) Preindex

STRB r0, [r1], #12



Figure 13.3 ARM Indexing Methods

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





Figure 13.4 ARM Load/Store Multiple Addressing

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

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

Allocation of Bits



	Opcode		D/I	Z/C			Di	splaceme	ent		
0	1.1	2	3	4	5		1		113.5		11
1	3 min		1. 36	1 million	10	i solo	in we		1. 36	Mr. W.	
al and	Contraction of the			Input	/Outpu	t Instru	ctions		1.	0 1	
1	1	0	2		De	vice		0	0	Opcode	
0		2	3		1.1.1			8	9		п.
				Register	Roforo	nco Inst	ructions	1.1			
Group	1 Microin	structio	ons	Register	Kelere	nee msi	1 uctions			14.1	
1	1	1	0	CLA	CLL	CMA	CML	RAR	RAL	BSW	IAC
0	1	2	3	4	5	6	7	8	9	10	11
1. 198	1 2 h HE	N' AN	a (* 1985)	274 ME	W and	1	24 46	Nº AND	1. 30	24 45	Nº KAL
Group	2 Microin	structio	ons		20 43	42.00			42.5		
1	1	1	1	CLA	SMA	SZA	SNL	RSS	OSR	HLT	0
0	1	2	3	4	5	6	7	8	9	10	11
			one	1 2 4 1	11		14 J	日本		14 1ª	1
Group	3 Microin	structio	5115		a statute of a			A STATISTICS AND A STATISTICS	A TOTAL OF A DESIGNATION OF		
Group 1	3 Microin 1	1	1	CLA	MQA	0	MQL	0	0	0	1
Group 1 0	3 Microin 1 1	1 2	1 3	CLA 4	MQA 5	0 6	MQL 7	0 8	0 9	0 10	1 11
Group 1 0	3 Microin 1 1	1 2	1 3	CLA 4	MQA 5	0 6	MQL 7	0 8	0 9	0 10	1 11
Group 1 0 D/I	3 Microin 1 = Direct//	1 2 Indirect	$\frac{1}{3}$ t addres	CLA 4 s	MQA 5	0 6 IAC =	MQL 7 = Incren	0 8 nent AC	0 9 cumulate	0 10 or	1 11
Group 1 0 D/I Z/C	<u>3 Microin</u> <u>1</u> = Direct/ = Page 0	1 2 Indirect	$\frac{1}{3}$ t addres	CLA 4 s e	MQA 5	0 6 IAC = SMA =	MQL 7 = Incren = Skip o	0 8 nent ACo on Minus	0 9 cumulate Accum	0 10 or ulator	<u>1</u> 11
Group 1 0 D/I Z/C CLA	3 Microin 1 = Direct/ = Page 0 = Clear A	1 2 Indirect or Curr	1 3 t addres rent pag lator	CLA 4 s e	MQA 5	0 6 IAC = SMA = SZA =	MQL 7 = Incren = Skip o = Skip o	0 8 nent ACo on Minus on Zero A	0 9 cumulate Accumul	0 10 or ulator ator	1
Group 1 0 D/I Z/C CLA CLL	3 Microin 1 = Direct/ = Page 0 = Clear A = Clear L	1 2 Indirect or Curr Accumu Link	1 3 t address rent pag lator	CLA 4 s e	MQA 5	0 6 IAC = SMA = SZA = SNL =	MQL 7 = Incren = Skip o = Skip o = Skip o	0 8 nent ACo on Minus on Zero A on Nonze	0 9 cumulate Accumul Accumul ero Link	0 10 or ulator ator	11
Group 1 0 D/I Z/C CLA CLL CMA	3 Microin 1 = Direct/I = Page 0 = Clear A = Clear L = CoMple	Indirect or Curr Accumu ink ement A	1 3 t addres rent pag lator Accumu	CLA 4 s e	MQA 5	0 6 IAC = SMA = SZA = SNL = RSS =	MQL 7 = Increm = Skip o = Skip o = Skip o = Rever	0 8 nent ACo on Minus on Zero A on Nonze se Skip S	0 9 cumulate Accumul cro Link Sense	0 10 or ulator ator	11
Group 1 0 D/I Z/C CLA CLL CMA CML	3 Microin 1 1 = Direct/I = Page 0 = Clear A = Clear L = CoMple = CoMple	Indirect or Curr Accumu Link ement I	1 3 rent pag lator Accumu Link	CLA 4 s e	MQA 5	0 6 SMA = SZA = SNL = RSS = OSR =	MQL 7 = Increm = Skip o = Skip o = Skip o = Revert = Or wit	0 8 nent ACo on Minus on Zero A on Nonze se Skip S th Switch	0 9 cumulate Accumul cro Link Sense n Registe	0 10 or ulator ator er	1
Group 1 0 D/I Z/C CLA CLL CMA CML RAR	3 Microin 1 1 = Direct/I = Page 0 = Clear A = Clear L = CoMple = Rotate	Indirect or Curr Accumu Link ement I Accum	1 3 t address rent pag lator Accumu Link ultator I	CLA 4 s e lator Right	MQA 5	0 6 SMA = SZA = SNL = RSS = OSR = HLT =	MQL 7 = Increm = Skip o = Skip o = Skip o = Rever = Or wit = HaLT	0 8 on Minus on Zero A on Nonze se Skip S th Switch	0 9 Accumulate Accumul ero Link Sense n Registe	0 10 or ulator ator er	11
Group 1 0 D/I Z/C CLA CLL CMA CML RAR RAR RAL	3 Microin 1 = Direct/I = Page 0 = Clear A = Clear L = CoMple = Rotate = Rotate	Indirect or Curr Accumu ink ement I Accum Accum	1 3 t addres rent pag lator Accumu Link ultator I ulator L	CLA 4 s e lator Right eft	MQA 5	0 6 SMA = SZA = SNL = RSS = OSR = HLT = MQA =	MQL 7 = Increm = Skip o = Skip o = Skip o = Rever = Or wit = HaLT = Multin	0 8 on Minus on Zero A on Nonze se Skip S th Switch	0 9 cumulate Accumul ero Link Sense n Registe	0 10 or ulator ator er er	1 11 nulator

Figure 13.5 PDP-8 Instruction Formats

Opcode 0 8	Register I 9 12	Index Register 14 17	Memory Address 35

I = indirect bit

Figure 13.6 PDP-10 Instruction Format

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



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

Figure 13.7 Instruction Formats for the PDP-11

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

Figure 13.8 Examples of VAX Instructions



Figure 13.9 x86 Instruction Format

	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	65	4	3 2 1 0
data processing immediate shift	cond	0	0	0	C	ppc	od	e	S	Rn	Rd	shift amou	nt	shift	0	Rm
data processing register shift	cond	0	0	0	C	рс	od	e	S	Rn	Rd	Rs	0	shift	1	Rm
data processing immediate	cond	0	0	1	0	рс	od	e	S	Rn	Rd	rotate		im	me	diate
load/store immediate offset	cond	0	1	0	Ρ	U	В	W	L	Rn	Rd	in	nm	ediate		
load/store register offset	cond	0	1	1	Ρ	U	В	W	L	Rn	Rd	shift amou	nt	shift	0	Rm
load/store multiple	cond	1	0	0	Ρ	U	S	W	L	Rn	and the second	regist	er	list	Sup.	
branch/branch with link	cond	1	0	1	L				-		24-bi	t 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

Figure 13.10 ARM Instruction Formats



Figure 13.11 Examples of Use of ARM Immediate Constants



Figure 13.12 Expanding a Thumb ADD Instruction into its ARM Equivalent

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 16bit 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



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

Figure 13.13 Thumb-2 Encoding

	NE COST DE LA CELE						
Addr	ess	Co	ntents		Address	Contents	
10	1 0010	0010	101	2201		101	2201
10	2 0001	0010	102	1202	1. 1. 10 13	102	1202
10	3 0001	0010	103	1203	1. 1. 1. S. S.	103	1203
10	4 0011	0010	104	3204		104	3204
20	1 0000	0000	201	0002		201	0002
20	2 0000	0000	202	0003	の時にはいた	202	0003
20	3 0000	0000	203	0004	all soft is	203	0004
20	4 0000	0000	204	0000	1. 1. 1. 1.	204	0000

(a) Binary program

(b) Hexadecimal program

Ş	A Constant	AND ALCO AND	Constant of	AT ALL AND AND	State of the state of the	and the second	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
	Address	Instru	iction		Label	Operation	Operand
100	101	LDA	201	Charles and the	FORMUL	LDA	Ι
1	102	ADD	202	and the second		ADD	J
1	103	ADD	203			ADD	Κ
-	104	STA	204	1224		STA	Ν
				S MEANER D.			
	201	DAT	2	the man and the	Ι	DATA	2
	202	DAT	3		J	DATA	3
100	203	DAT	4	14512 世际 八	K	DATA	4
1	204	DAT	0		N	DATA	0

(c) Symbolic program

(d) Assembly program

Figure 13.14 Computation of the Formula N = I + J+ K

Summary

Chapter 13

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

Instruction Sets: Addressing Modes and Formats

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