Microprocessadores e Microcontroladores (27146)

Conjunto de instruções (x86)

Prof. Ricardo Menotti (menotti@ufscar.br)

Atualizado em: 26 de abril de 2021

Departamento de Computação
Centro de Ciências Exatas e de Tecnologia
Universidade Federal de São Carlos

Processadores Intel

- 8086/8088 (16 bits) introduzidos no final da década de 1970 (3 µm);
- 80386 DX/SX (32 bits) introduzidos em 1989 (1 µm);
- Pentium (32 bits) com instruções MMX introduzido em 1997 (0.35 µm);
- Xeon (64 bits) compativel com AMD64 introduzido em 2004 (90 nm);
- Core i3/i5/i7/i9 hoje na 10ª geração (14 nm);

Principais diferenças do x86 com ARM/RISC [Harris and Harris(2016)]

Table 6.19 Major differences between ARM and x86

Feature	ARM	x86
# of registers	15 general purpose	8, some restrictions on purpose
# of operands	3–4 (2–3 sources, 1 destination)	2 (1 source, 1 source/destination)
operand location	registers or immediates	registers, immediates, or memory
operand size	32 bits	8, 16, or 32 bits
condition flags	yes	yes
instruction types	simple	simple and complicated
instruction encoding	fixed, 4 bytes	variable, 1–15 bytes

Operandos em memória [Harris and Harris(2016)]

Table 6.20 Operand locations

Source/ Destination	Source	Example	Meaning
register	register	`add EAX, EBX`	`EAX <– EAX + EBX`
register	immediate	`add EAX, 42`	`EAX <– EAX + 42`
register	memory	`add EAX, [20]`	`EAX <– EAX + Mem[20]`
memory	register	`add [20], EAX`	`Mem[20] <– Mem[20] + EAX`
memory	immediate	`add [20], 42`	`Mem[20] <– Mem[20] + 42`

Modos de endereçamento [Harris and Harris(2016)]

Table 6.21 Memory addressing modes

Example	Meaning	Comment
`add EAX, [20]`	`EAX <– EAX + Mem[20]`	displacement
`add EAX, [ESP]`	`EAX <– EAX + Mem[ESP]`	base addressing
`add EAX, [EDX+40]`	`EAX <– EAX + Mem[EDX+40]`	base + displacement
`add EAX, [60+EDI*4]`	`EAX <– EAX + Mem[60+EDI*4]`	displacement + scaled index
`add EAX, [EDX+80+EDI*2]`	`EAX <– EAX + Mem[EDX+80+EDI*2]`	base + displacement + scaled index

Table 6.22 Instructions acting on 8-, 16-, or 32-bit data

Example	Meaning	Data Size
`add AH, BL`	`AH <– AH + BL`	8-bit
`add AX, –1`	`AX <– AX + 0xFFFF`	16-bit
`add EAX, EDX`	`EAX <– EAX + EDX`	32-bit

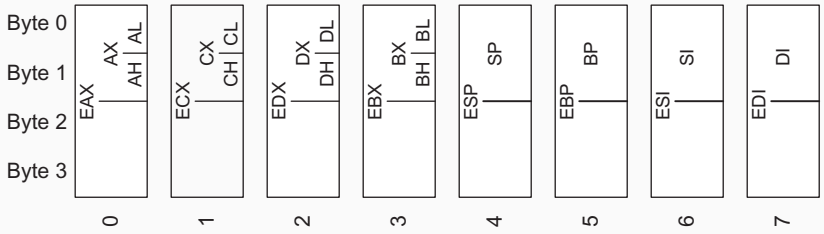


Figure 6.35 x86 registers

Algumas instruções (i) [Harris and Harris(2016)]

Table 6.24 Selected x86 instructions

Instruction	Meaning	Function
ADD/SUB	add/subtract	D = D + S / D = D − S
ADDC	add with carry	D = D + S + CF
INC/DEC	increment/decrement	D = D + 1 / D = D − 1
CMP	compare	Set flags based on D − S
NEG	negate	D = − D
AND/OR/XOR	logical AND/OR/XOR	D = D op S
NOT	logical NOT	$D = \overline{D}$
IMUL/MUL	signed/unsigned multiply	EDX:EAX = EAX × D
IDIV/DIV	signed/unsigned divide	EDX:EAX/D EAX = Quotient; EDX = Remainder
SAR/SHR	arithmetic/logical shift right	D = D >>> S / D = D >> S
SAL/SHL	left shift	D = D << S
ROR/ROL	rotate right/left	Rotate D by S
RCR/RCL	rotate right/left with carry	Rotate CF and D by S

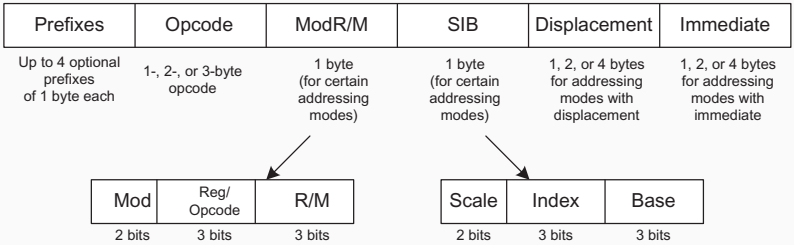
Algumas instruções (ii) [Harris and Harris(2016)]

BT	bit test	CF = D[S] (the S*th* bit of D)
BTR/BTS	bit test and reset/set	CF = D[S]; D[S] = 0 / 1
TEST	set flags based on masked bits	Set flags based on D AND S
MOV	move	D = S
PUSH	push onto stack	ESP = ESP −4; Mem[ESP] = S
POP	pop off stack	D = MEM[ESP]; ESP = ESP + 4
CLC, STC	clear/set carry flag	CF = 0 / 1
JMP	unconditional jump	relative jump: EIP = EIP + S absolute jump: EIP = S
Jcc	conditional jump	if (flag) EIP = EIP + S
LOOP	loop	ECX = ECX −1 if (ECX ≠ 0) EIP = EIP + imm
CALL	function call	ESP = ESP −4; MEM[ESP] = EIP; EIP = S
RET	function return	EIP = MEM[ESP]; ESP = ESP + 4

Saltos [Harris and Harris(2016)]

Instruction	Meaning	Function after `CMP D, S`
`JZ/JE`	jump if `ZF = 1`	jump if `D = S`
`JNZ/JNE`	jump if `ZF = 0`	jump if `D ≠ S`
`JGE`	jump if `SF = OF`	jump if `D ≥ S`
`JG`	jump if `SF = OF` and `ZF = 0`	jump if `D >S`
`JLE`	jump if `SF ≠ OF` or `ZF = 1`	jump if `D ≤ S`
`JL`	jump if `SF ≠ OF`	jump if `D <S`
`JC/JB`	jump if `CF = 1`	
`JNC`	jump if `CF = 0`	
`JO`	jump if `OF = 1`	
`JNO`	jump if `OF = 0`	
`JS`	jump if `SF = 1`	
`JNS`	jump if `SF = 0`	

Codificação das instruções [Harris and Harris(2016)]



Para saber mais e praticar...

- http://ref.x86asm.net/index.html
- https://carlosrafaelgn.com.br/Asm86/
- https://www.tutorialspoint.com/compile_assembly_online.php
- https://software.intel.com/sites/landingpage/IntrinsicsGuide/

Bibliografia


David Harris and Sarah Harris.
Digital Design and Computer Architecture: ARM® Edition.
Morgan Kaufmann, 2016.