Instruções assembly da família Intel x86-64 - Sintaxe AT&T

Noemi Rodriguez, Ana Lúcia de Moura, Raúl Renteria, Alexandre Meslin

18 de Abril de 2021

Template

```
.data
label:
    .<tipo>   <valores separados por virgula>

.text
.globl  main
main:

/*************************************************************/
/* mantenha este trecho aqui - prologo!!!                    */
  pushq   %rbp
  movq    %rsp, %rbp
  subq    $16, %rsp
  movq    %rbx, -8(%rbp)
  movq    %r12, -16(%rbp)
/*************************************************************/


/*************************************************************/
/* seu codigo aqui                                           */
/*************************************************************/


/*************************************************************/
/* mantenha este trecho aqui - finalizacao!!!!               */
  movq  $0, %rax  // rax = 0  (valor de retorno)
  movq  -8(%rbp), %rbx
  movq  -16(%rbp), %r12
  leave
  ret
/*************************************************************/
```

Registradores

Registradores Inteiros

64 bits	32 bits	16 bits	8 bits		Obs.
%rax	%eax	%ax	%ah	%al	valor de retorno
%rbx	%ebx	%bx	%bh	%bl	callee-saved
%rcx	%ecx	%cx	%ch	%cl	4° argumento
%rdx	%edx	%dx	%dh	%dl	3° argumento
%rsi	%esi	%si		%sil	2° argumento
%rdi	%edi	%di		%dil	1° argumento
%r8	%r8d	%r8w		%r8b	5° argumento
%r9	%r9d	%r9w		%r9b	6° argumento
%r10	%r10d	%r10w		%r10b	
%r11	%r11d	%r11w		%r11b	
%r12	%r12d	%r12w		%r12b	callee-saved
%r13	%r13d	%r13w		%r13b	callee-saved
%r14	%r14d	%r14w		%r14b	callee-saved
%r15	%r15d	%r15w		%r15b	callee-saved
%rbp					frame pointer
%rsp					stack pointer

Registrador de Flags

Bit #	Abrev.	Descrição	
0	CF	Carry flag	FLAGS
1	Reservado, always 1 in EFLAGS		
2	PF	Parity flag	
3	Reservado		
4	AF	Adjust flag	
5		Reservado	
6	ZF	Zero flag	
7	SF	Sign flag	
8	TF	Trap flag (single step)	
9	IF	Interrupt enable flag	
10	DF	Direction flag	
11	OF	Overflow flag	
12-13	IOPL	I/O privilege level (286+ only), always 1 on 8086 and 186	
14	NT	Nested task flag (286+ only), always 1 on 8086 and 186	
15	Reservado, always 1 on 8086 and 186, always 0 on later models		
16	RF	Resume flag (386+ only)	EFLAGS
17	VM	Virtual 8086 mode flag (386+ only)	
18	AC	Alignment check (486SX+ only)	
19	VIF	Virtual interrupt flag (Pentium+)	
20	VIP	Virtual interrupt pending (Pentium+)	
21	ID	Able to use CPUID instruction (Pentium+)	
22-31	Reservado		
32-63	Reservado		RFLAGS

Seções do Programa

Dados (.data)

Instruções (.text)

- Todas as instruções contendo mais de um operando, guardam seu resultado em op2.
- Para especificar o tamanho do(s) operando(s), acrescentar sufixo na instrução: **B,W,L** ou **Q**.

Nome	Descrição
.ascii	Text string
.asciz	Null-terminated text string
.string	Null-terminated text string
.byte	Byte value
.short	16-bit integer number
.int	32-bit integer number
.long	32-bit integer number (same as .int)
.quad	8-byte integer number
.octa	16-byte integer number
.float	Single-precision floating-point number
.single	Single-precision floating-point number (same as .float)
.double	Double-precision floating-point number

Movimentação

MOV	op1	op2	copia para op2 o valor de op1
MOVABSQ	op1	op2	copia para op2 (64 bits) o valor de op1 (constante de 64 bits)
PUSHQ	op1		atualiza o SP e coloca op1 no topo da pilha
POPQ	op1		transfere o topo da pilha para op1 e atualiza o seu topo

Operações aritméticas binárias

ADD	op1	op2	guarda em op2 o resultado de op2 + op1
SUB	op1	op2	guarda em op2 o resultado de op2 - op1
IMUL	op1	op2	guarda em op2 resultado de op1 * op2

Operações aritméticas unárias

INC	op1		incrementa o valor de op1
DEC	op1		decrementa o valor de op1
NEG	op1		complemento a 2 do valor de op1

Operações lógicas

AND	op1	op2	guarda em op2 o resultado de op1 `&` op2
OR	op1	op2	guarda em op2 o resultado de op1 `|` op2
XOR	op1	op2	guarda em op2 o resultado de op1 ^ op2
NOT	op1		guarda em op1 o resultado de ~op1

Comparação – atualizam as flags

CMP	op1	op2	atualiza flags com op2 - op1
TEST	op1	op2	atualiza flags conforme resultado de op2 AND op1 (não altera op2)

Operações de deslocamento

SHL	constante	op 1	desloca op1 o número de bits indicado para a esquerda
SHR	constante	op 1	desloca op1 o número de bits indicado para a direita (shift lógico)
SAR	constante	op 1	desloca o número de bits indicado para a direita (shift aritmético)

Cálculo de indireção (não acessa a memória

LEAQ	op1	op2	guarda em op2 o ENDEREÇO de op1 ("load effective address")

Extensão considerando sinal

MOVSB[W][L][Q]	op1	op2	guarda em op2 (w,l,q) o valor estendido de op1 (8 bits)
MOVSW[L][Q]	op1	op2	guarda em op2 (l,q) o valor estendido de op1 (16 bits)
MOVSLQ	op1	op2	guarda em op2 (64 bits) o valor estendido de op1 (32 bits)

Extensão sem sinal

MOVZB[W][L][Q]	op1	op2	guarda em op2 (w,l,q) o valor estendido de op1 (8 bits)

MOVZW[L][Q]	op1	op2	guarda em op2 (l,q) o valor estendido de op1 (16 bits)

Desvio condicional

JE/JZ	label	transfere para label caso o flag ZF (resultado zero) esteja setado
JNE/JNZ	label	transfere para label caso o flag ZF não esteja setado

Desvio condicional para operações sem sinal

JA/JNBE	label	transfere se superior ("jump if above")
JAE/JNB	label	transfere se superior ou igual ("jump if above or equal")
JB/JNAE	label	transfere se inferior ("jump if below")
JBE/JNA	label	transfere se inferior ou igual ("jump if below or equal")

Desvio condicional considerando sinal

JG/JNLE	label	transfere se superior ("jump if greater")
JGE/JNL	label	transfere se superior ou igual ("jump if greater or equal")
JL/JNGE	label	transfere se inferior ("jump if less")
JLE/JNG	label	transfere se inferior ou igual ("jump if less or equal")

Desvio incondicional

JMP	label	transfere para label
CALL	label	transfere para label, empilhando endereço de retorno
RET		transfere para endereço retirado do topo da pilha
INT	valor	transfere para o tratador da interrupção indicada

Instruções de Ponto Flutuante

Registradores: %xmm0 a %xmm15
Argumentos: %xmm0 a %xmm7
Valor de Retorno: %xmm0

Movimentação e Conversão de Dados

MOVS[S][D]	op1	op2	copia para op2 o valor de op1
CVTSS2SD	op1	op2	converte float em op1 para double em op2 (reg)
CVTSD2SS	op1	op2	converte double em op1 para float em op2 (reg)
CVTSI2SS	op1	op2	converte inteiro em op1 para float em op2 (reg)
CVTSI2SD	op1	op2	converte inteiro em op1 para double em op2 (reg)
CVTTSS2SI	op1	op2	converte float em op1 para inteiro em op2 (reg)
CVTTSD2SI	op1	op2	converte double em op1 para inteiro em op2 (reg)
CVTTSS2SIQ	op1	op2	converte float em op1 para long em op2 (reg)
CVTTSD2SIQ	op1	op2	converte double em op1 para long em op2 (reg)

Operações Aritméticas

ADDS[S][D]	op1	op2	guarda em op2 (reg) o resultado de op2 + op1
SUBS[S][D]	op1	op2	guarda em op2 (reg) o resultado de op2 - op1
MULS[S][D]	op1	op2	guarda em op2 (reg) o resultado de op2 * op1
DIVS[S][D]	op1	op2	guarda em op2 (reg) o resultado de op2 / op1

Operações lógicas

PAND	op1	op2	guarda em op2 (reg) o resultado de op1 AND op2
POR	op1	op2	guarda em op2 (reg) o resultado de op1 OR op2
PXOR	op1	op2	guarda em op2 (reg) o resultado de op1 XOR op2
PANDN	op1	op2	guarda em op2 (reg) o resultado de NOT (op1 AND op2)

Comparação

UCOMIS[S][D]	op1	op2	compara op2 com op1 (testar resultado com condições de comparação sem sinal: a, ae, b, be)