**AT&T指令集**

GAS中每个操作都是有一个字符的后缀，表明操作数的大小。

|  |  |  |
| --- | --- | --- |
| C声明 | GAS后缀 | 大小(字节) |
| char | b | 1 |
| short | w | 2 |
| (unsigned) int / long / char\* | l | 4 |
| float | s | 4 |
| double | l | 8 |
| long double | t | 10/12 |

注意：GAL使用后缀“l”同时表示4字节整数和8字节双精度浮点数，这不会产生歧义因为浮点数使用的是完全不同的指令和寄存器。

**操作数格式：**

|  |  |  |  |
| --- | --- | --- | --- |
| 格式 | 操作数值 | 名称 | 样例（GAS = C语言） |
| $Imm | Imm | 立即数寻址 | $1 = 1 |
| Ea | R[Ea] | 寄存器寻址 | %eax = eax |
| Imm | M[Imm] | 绝对寻址 | 0x104 = \*0x104 |
| （Ea） | M[R[Ea]] | 间接寻址 | （%eax）= \*eax |
| Imm(Ea) | M[Imm+R[Ea]] | (基址+偏移量)寻址 | 4(%eax) = \*(4+eax) |
| （Ea,Eb） | M[R[Ea]+R[Eb]] | 变址 | (%eax,%ebx) = \*(eax+ebx) |
| Imm（Ea,Eb） | M[Imm+R[Ea]+R[Eb]] | 寻址 | 9(%eax,%ebx)= \*(9+eax+ebx) |
| (,Ea,s) | M[R[Ea]\*s] | 伸缩化变址寻址 | (,%eax,4)= \*(eax\*4) |
| Imm(,Ea,s) | M[Imm+R[Ea]\*s] | 伸缩化变址寻址 | 0xfc(,%eax,4)= \*(0xfc+eax\*4) |
| (Ea,Eb,s) | M(R[Ea]+R[Eb]\*s) | 伸缩化变址寻址 | (%eax,%ebx,4) = \*(eax+ebx\*4) |
| Imm(Ea,Eb,s) | M(Imm+R[Ea]+R[Eb]\*s) | 伸缩化变址寻址 | 8(%eax,%ebx,4) = \*(8+eax+ebx\*4) |

注：M[xx]表示在存储器中xx地址的值，R[xx]表示寄存器xx的值，这种表示方法将寄存器、内存都看出一个大数组的形式。

**数据传送指令：**

|  |  |  |
| --- | --- | --- |
| 指令 | 效果 | 描述 |
| movl S,D | D <-- S | 传双字 |
| movw S,D | D <-- S | 传字 |
| movb S,D | D <-- S | 传字节 |
| movsbl S,D | D <-- 符号扩展S | 符号位填充(字节->双字) |
| movzbl S,D | D <-- 零扩展S | 零填充(字节->双字) |
| pushl S | R[%esp] <-- R[%esp] – 4;  M[R[%esp]] <-- S | 压栈 |
| popl D | D <-- M[R[%esp]]；  R[%esp] <-- R[%esp] + 4; | 出栈 |

注：均假设栈往低地址扩展。

**算数和逻辑操作地址：**

|  |  |  |
| --- | --- | --- |
| 指令 | 效果 | 描述 |
| leal S,D | D = &S | movl改版，S地址入D，D仅能是寄存器 |
| incl D | D++ | 加1 |
| decl D | D-- | 减1 |
| negl D | D = -D | 取负 |
| notl D | D = ~D | 取反 |
| addl S,D | D = D + S | 加 |
| subl S,D | D = D – S | 减 |
| imull S,D | D = D\*S | 乘 |
| xorl S,D | D = D ^ S | 异或 |
| orl S,D | D = D | S | 或 |
| andl S,D | D = D & S | 与 |
| sall k,D | D = D << k | 左移 |
| shll k,D | D = D << k | 左移(同sall) |
| sarl k,D | D = D >> k | 算数右移 |
| shrl k,D | D = D >> k | 逻辑右移 |

**特殊算术操作：**

|  |  |  |
| --- | --- | --- |
| 指令 | 效果 | 描述 |
| imull S | R[%edx]:R[%eax] = S \* R[%eax] | 无符号64位乘 |
| mull S | R[%edx]:R[%eax] = S \* R[%eax] | 有符号64位乘 |
| cltd S | R[%edx]:R[%eax] = 符号位扩展R[%eax] | 转换为4字节 |
| idivl S | R[%edx] = R[%edx]:R[%eax] % S;  R[%eax] = R[%edx]:R[%eax] / S; | 有符号除法，保存余数和商 |
| divl S | R[%edx] = R[%edx]:R[%eax] % S;  R[%eax] = R[%edx]:R[%eax] / S; | 无符号除法，保存余数和商 |

注：64位数通常存储为，高32位放在edx，低32位放在eax。

**条件码：**

条件码寄存器描述了最近的算数或逻辑操作的属性。

CF：进位标志，最高位产生了进位，可用于检查无符号数溢出。

OF：溢出标志，二进制补码溢出——正溢出或负溢出。

ZF：零标志，结果为0。

SF：符号标志，操作结果为负。

**比较指令：**

|  |  |  |
| --- | --- | --- |
| 指令 | 基于 | 描述 |
| cmpb S2,S1 | S1 – S2 | 比较字节，差关系 |
| testb S2,S1 | S1 & S2 | 测试字节，与关系 |
| cmpw S2,S1 | S1 – S2 | 比较字，差关系 |
| testw S2,S1 | S1 & S2 | 测试字，与关系 |
| cmpl S2,S1 | S1 – S2 | 比较双字，差关系 |
| testl S2,S1 | S1 & S2 | 测试双字，与关系 |

**访问条件码指令：**

|  |  |  |  |
| --- | --- | --- | --- |
| 指令 | 同义名 | 效果 | 设置条件 |
| sete D | setz | D = ZF | 相等/零 |
| setne D | setnz | D = ~ZF | 不等/非零 |
| sets D |  | D = SF | 负数 |
| setns D |  | D = ~SF | 非负数 |
| setg D | setnle | D = ~(SF ^OF) & ZF | 大于（有符号>） |
| setge D | setnl | D = ~(SF ^OF) | 小于等于(有符号>=) |
| setl D | setnge | D = SF ^ OF | 小于(有符号<) |
| setle D | setng | D = (SF ^ OF) | ZF | 小于等于(有符号<=) |
| seta D | setnbe | D = ~CF & ~ZF | 超过(无符号>) |
| setae D | setnb | D = ~CF | 超过或等于(无符号>=) |
| setb D | setnae | D = CF | 低于(无符号<) |
| setbe D | setna | D = CF | ZF | 低于或等于(无符号<=) |

**跳转指令：**

|  |  |  |  |
| --- | --- | --- | --- |
| 指令 | 同义名 | 跳转条件 | 描述 |
| jmp   Label |  | 1 | 直接跳转 |
| jmp   \*Operand |  | 1 | 间接跳转 |
| je     Label | jz | ZF | 等于/零 |
| jne    Label | jnz | ~ZF | 不等/非零 |
| js     Label |  | SF | 负数 |
| jnz    Label |  | ~SF | 非负数 |
| jg     Label | jnle | ~(SF^OF) & ~ZF | 大于(有符号>) |
| jge    Label | jnl | ~(SF ^ OF) | 大于等于(有符号>=) |
| jl     Label | jnge | SF ^ OF | 小于（有符号<） |
| jle     Label | jng | (SF ^ OF) | ZF | 小于等于(有符号<=) |
| ja     Label | jnbe | ~CF & ~ZF | 超过(无符号>) |
| jae    Label | jnb | ~CF | 超过或等于(无符号>=) |
| jb     Label | jnae | CF | 低于(无符号<) |
| jbe    Label | jna | CF | ZF | 低于或等于(无符号<=) |

**转移控制指令：（函数调用）：**

|  |  |
| --- | --- |
| 指令 | 描述 |
| call    Label | 过程调用，返回地址入栈，跳转到调用过程起始处，返回地址是call后面那条指令的地址 |
| call    \*Operand |
| leave | 为返回准备好栈，为ret准备好栈，主要是弹出函数内的栈使用及%ebp |

**IA-32处理器常用整数指令分类一览**

     指令符号说明

|  |  |
| --- | --- |
| 符号 | 说明 |
| r8 | 任意一个8位通用寄存器AH/AL/BH/BL/CH/CL/DH/DL |
| r16 | 任意一个16位通用寄存器AX/BX/CX/DX/SI/DI/BP/SP |
| r32 | 任意一个32位通用寄存器EAX/EBX/ECX/EDX/ESI/EDI/EBP/ESP |
| reg | 代表r8/r16/r32 |
| seg | 段寄存器CS/DS/ES/SS和FS/GS |
| m8 | 一个8位存储器操作数单元 |
| m16 | 一个16位存储器操作数单元 |
| m32 | 一个32位存储器操作数单元 |
| mem | 代表m8/m16/m32 |
| i8 | 一个8位立即数 |
| i16 | 一个16位立即数 |
| i32 | 一个32位立即数 |
| imm | 代表i8/i16/i32 |
| dest | 目的操作数 |
| src | 源操作数 |
| label | 标号 |
| m64 | 一个64位存储器操作数单元 |

     IA-32常用指令的汇编格式（AT&T语法）

|  |  |  |
| --- | --- | --- |
| 指令类型 | 指令汇编格式 | 指令功能简介 |
|  | MOVimm, reg/mem | dest←src |
| 传送指令 | MOV reg , reg/mem/seg |  |
|  | MOV mem, reg/seg |  |
|  | MOV seg, reg/mem |  |
|  | CMOVxx r16/m16, r16 | 条件成立，r16←r16/m16 |
|  | CMOVxx r32/m32, r32 | 条件成立，r32←r32/m32 |
| 交换指令 | XCHG reg, reg/mem | reg←→reg/mem |
|  | XCHG reg/mem, reg |  |
|  | BSWAP r32 | 字节交换 |
|  | XADD reg, reg/mem | 交换加 |
|  | CMPXCHG reg/mem,reg | 比较交换 |
|  | CMPXCHG8B m64 | 8字节比较交换 |
|  | PUSH reg/mem/seg | 寄存器/存储器入栈（只支持16、32位数据） |
|  | PUSH imm | 立即数入栈（只支持16、32位数据） |
|  | POP reg/seg/mem | 出栈（只支持16、32位数据） |
| 堆栈指令 | PUSHA | 保护所有r16 |
|  | POPA | 恢复所有r16 |
|  | PUSHAL | 保护所有r32 |
|  | POPAL | 恢复所有r32 |
| 标志传送 | PUSHFL | EFLAGS入栈 |
|  | POPFL | EFLAGS出栈 |
| 地址传送 | LEA mem, r16/r32 | r16/r32←16/32位有效地址 |
|  | ADD imm/reg/mem, reg | dest←dest＋src |
|  | ADD imm/reg, mem |  |
| 加法运算 | ADC imm/reg/mem, reg | dest←dest＋src＋CF |
|  | ADC imm/reg, mem |  |
|  | INC reg/mem | reg/mem←reg/mem＋1 |
|  | SUB imm/reg/mem, reg | dest←dest－src |
|  | SUB imm/reg, mem |  |
|  | SBB imm/reg/mem, reg | dest←dest－src－CF |
| 减法运算 | SBB imm/reg, mem |  |
|  | DEC reg/mem | reg/mem←reg/mem－1 |
|  | NEG reg/mem | reg/mem←0－reg/mem |
|  | CMP imm/reg/mem, reg | dest－src |
|  | CMP imm/reg, mem |  |
|  | MUL reg/mem | 无符号数值乘法 |
|  | IMUL reg/mem | 有符号数值乘法 |
| 乘法运算 | IMUL r16,r16/m16/i8/i16 | r16←r16×r16/m16/i8/i16 |
|  | IMUL r16,r/m16,i8/i16 | r16←r/m16×i8/i16 |
|  | IMUL r32,r32/m32/i8/i32 | r32←r32×r32/m32/i8/i32 |
|  | IMUL r32,r32/m32,i8/i32 | r32←r32/m32×i8/i32 |
| 除法运算 | DIV reg/mem | 无符号数值除法 |
|  | IDIV reg/mem | 有符号数值除法 |
|  | CLTD | 把EAX符号扩展为EDX.EAX |
|  | MOVSX r8/m8, r16 | 把r8/m8符号扩展并传送至r16 |
| 符号扩展 | MOVSX r8/m8/r16/m16, r32 | 把r8/m8/r16/m16符号扩展并传送至r32 |
|  | MOVZX r8/m8/r16/m16, r32 | 把r8/m8零位扩展并传送至r16 |
|  | MOVZX r32,r8/m8/r16/m16 | 把r8/m8/r16/m16零位扩展并传送至r32 |
|  | DAA | 将AL中的加和调整为压缩BCD码 |
|  | DAS | 将AL中的减差调整为压缩BCD码 |
| 十进制调整 | AAA | 将AL中的加和调整为非压缩BCD码 |
|  | AAS | 将AL中的减差调整为非压缩BCD码 |
|  | AAM | 将AX中的乘积调整为非压缩BCD码 |
|  | AAD | 将AX中的非压缩BCD码扩展成二进制数 |
|  | AND imm/reg/mem, reg | dest←dest AND src |
|  | AND imm/reg, mem |  |
|  | OR imm/reg/mem, reg | dest←dest OR src |
| 逻辑运算 | OR imm/reg, mem |  |
|  | XOR imm/reg/mem, reg | dest←dest XOR src |
|  | XOR imm/reg, mem |  |
|  | TEST imm/reg/mem, reg | dest AND src |
|  | TEST imm/reg, mem |  |
|  | NOT reg/mem | reg/mem←NOT reg/mem |
|  | SAL 1/%cl/i8, reg/mem | 算术左移1/CL/i8指定的次数 |
| 移位 | SAR 1/%cl/i8, reg/mem | 算术右移1/CL/i8指定的次数 |
|  | SHL 1/%cl/i8, reg/mem | 逻辑左移（与SAL相同） |
|  | SHR 1/%cl/i8, reg/mem | 逻辑右移1/CL/i8指定的次数 |
|  | ROL 1/%cl/i8, reg/mem | 循环左移1/CL/i8指定的次数 |
| 循环移位 | ROR 1/%cl/i8, reg/mem | 循环右移1/CL/i8指定的次数 |
|  | RCL 1/%cl/i8, reg/mem | 带进位循环左移1/CL/i8指定的次数 |
|  | RCR 1/%cl/i8, reg/mem | 带进位循环右移1/CL/i8指定的次数 |
|  | MOVS[B/W/L] | 串传送 |
|  | LODS[B/W/L] | 串读取 |
|  | STOS[B/W/L] | 串存储 |
| 串操作 | CMPS[B/W/L] | 串比较 |
|  | SCAS[B/W/L] | 串扫描 |
|  | REP | 重复前缀 |
|  | REPZ / REPE | 相等重复前缀 |
|  | REPNZ / REPNE | 不等重复前缀 |
|  | JMP label | 无条件直接转移 |
|  | JMP r16/r32/m16 | 无条件间接转移 |
| 转移 | Jxx label | 条件转移 |
|  | JCXZ label | CX等于0转移 |
|  | JECXZ label | ECX等于0转移 |
|  | LOOP label | ECX←ECX－1；若ECX≠0，循环 |
| 循环 | LOOPZ / LOOPE label | ECX←ECX－1；若ECX≠0且ZF=1，循环 |
|  | LOOPNZ / LOOPNE label | ECX←ECX－1；若CX≠0且ZF=0，循环 |
|  | CALL label | 直接调用 |
| 子程序 | CALL r16/m16 | 间接调用 |
|  | RET | 无参数返回 |
|  | RET i16 | 有参数返回 |
|  | INT i8 | 中断调用 |
| 中断 | IRET | 中断返回 |
|  | INTO | 溢出中断调用 |
| 高级语言支持 | ENTER i16,i8 | 建立堆栈帧 |
|  | LEAVE | 释放堆栈帧 |
|  | CLC | CF←0 |
|  | STC | CF←1 |
|  | CMC | CF←~CF |
| 处理器控制 | CLD | DF←0 |
|  | STD | DF←1 |
|  | NOP | 空操作指令 |
|  | CPUID | 返回处理器的有关特征信息 |

AT&T汇编enter指令和leave指令

enter指令

在AT&T汇编中，enter等效于以下汇编指令：

pushl %ebp # 将%ebp压栈

movl %esp %ebp # 将%esp保存到%ebp， 这两步是函数的标准开头

leave指令

在AT&T汇编中，leave等效于以下汇编指令：

movl %ebp, %esp

popl %ebp

call指令

在AT&T汇编中，call foo（foo是一个标号）等效于以下汇编指令：

pushl %eip

movl f, %eip

ret指令

在AT&T汇编中，ret等效于以下汇编指令：

popl %eip