

# La machine TAM

---

## 1. TAM 地址

TAM 地址的格式为  $d[r]$ ，其中  $d$  是整数， $r$  是寄存器的名称。

- SB: 栈底。
- ST: 栈顶。堆栈顶部的数据始终位于  $-1[ST]$
- HB: 堆底
- HT: 堆顶。堆顶的数据总是  $-1[HT]$ 。
- LB: 指向当前的寄存器。
  - $0[LB]$  包含静态链接（对于 Microjava，始终为 0）
  - $1[LB]$  包含从函数返回时执行的指令，由 CALL 或 CALLI 自动分配。
  - $2[LB]$  包含 LB 的旧值（调用函数的基数），由 CALL 或 CALLI 自动分配。
- CB: 代码底部
- CT: 代码顶部
- CP: 当前指令

## 2. TAM 指令

其中  $n$  和  $d$  是整数， $r$  是寄存器名称

- LOADL  $c$ : 将常量  $c$ （数字或字符串）入栈。（真=1，假=0）
- LOAD  $(n) d[r]$ : 将  $n$  的副本入栈，地址为  $d[r]$
- LOADA  $d[r]$  or  $label$ : 将地址  $d[r]$  或  $label$  入栈
- LOADI  $(n)$ : 将  $n$  的副本入栈顶
- STORE  $(n) d[r]$ : 复制栈顶地址  $d[r]$  处的  $n$ 。将  $n$  出栈。
- STOREI  $(n)$ : 将从栈顶取出的  $n$  个字复制到栈顶的地址。弹出地址和  $n$ 。
- PUSH  $n$ : 将 ST 向上移动  $n$  个位置,  $ST = ST + n$
- POP  $(d) n$ : 将 ST 向下移动  $n$  个位置并重新将位于堆栈顶部的  $d$  个单词入栈。
- JUMP  $label$  或 JUMP  $d[r]$ : 跳转到  $label$  或地址 ( $r = CB、CT、CP$ )。
- JUMPIF( $n$ )  $label$  或 JUMPIF( $n$ )  $d[r]$ : 如果栈顶的值等于  $n$ , 跳转到  $label$  或地址 ( $r = CB、CT、CP$ )
- JUMPI: 跳转到栈顶的地址。
- CALL (LB)  $label$  或 CALL (LB)  $d[r]$ : 在  $label$  或地址  $d[r]$  处调用函数 ( $r = CB、CT、CP$ )。
- CALLI: 调用到栈顶地址的函数（警告：您必须先入栈一个 0，然后是地址）
- RETURN  $(n) d$ : 函数的返回，通过删除参数的  $d$  个字并重新编译返回值的  $n$  个字。
- SUBR  $prim$ : 对原语(primitive)的调用。
- HALT: 停止程序

### 3. TAM 原语 (primitive)

Boolean:

名称	参数	结果	描述
BNeg	1	1	逻辑非
BOr	2	1	逻辑或
BAnd	2	1	逻辑与
BIn	0	1	从 stdin 读取一个 bool 值 (1 或 0)
BOut	1	0	向 stdout 显示一个 bool 值 (true 或 false)
B2C	1	1	将 Bool 转换为 字符, true = '1', false = '0'
B2I	1	1	将 Bool 转换为 int, true = 1, false = 0
B2S	1	1	将 Bool 转换为 字符串 (将字符串的地址压栈在字符串数组中)

字符 Char:

名称	参数	结果	描述
COut	1	0	向 stdout 显示一个字符
CIn	0	1	从 stdin 读取一个字符
C2B	1	1	将 字符 转换为 Bool, '1' = true, '0' = false
C2I	1	1	将 字符 转换为 int (ASCII)
C2S	1	1	将 字符 转换为 字符串 (将字符串的地址压栈在字符串数组中)

整型 int i:

名称	参数	结果	描述
INeg	1	1	$i = -i$
IAdd	2	1	$i = i_1 + i_2$
ISub	2	1	$i = i_1 - i_2$
IMul	2	1	$i = i_1 \times i_2$
IDiv	2	1	$i = i_1 \div i_2$
IMod	2	1	$i = i_1 \bmod i_2$
IEq	2	1	$i_1 = i_2 ?$
INeq	2	1	$i_1 \neq i_2 ?$
ILss	2	1	$i_1 < i_2 ?$
ILeq	2	1	$i_1 \leq i_2 ?$
IGtr	2	1	$i_1 > i_2 ?$
IGeq	2	1	$i_1 \geq i_2 ?$
IOut	1	0	向 stdout 显示一个 int 值
IIn	0	1	从 stdin 读取一个 int 值
I2B	1	1	将 int 转换为 bool, 1 = true, 0 = false
I2C	1	1	将 int 转换为 字符 (ASCII)
I2S	1	1	将 int 转换为 字符串 (将字符串的地址压栈在字符串数组中)

内存 Memory:

名称	参数	结果	描述
MVoid	0	1	返回未初始化的地址值
MAlloc	1	1	分配一个内存块并返回它的地址
MFree	1	0	释放内存块
MCompare	2	1	测试 地址位于堆栈顶部的 2 个内存块的内容之间的相等性
MCopy	3	0	将一个内存块的内容复制到第二个内存块中: size, @_destination, @_source

字符串 String:

名称	参数	结果	描述
SAlloc	1	1	创建一个空字符串并将其地址入栈字符串数组
SFree	1	0	释放字符串空间
SCopy	1	1	复制字符串并将其地址入栈字符串数组
SConcat	2	1	连接两个字符串
SOut	1	0	在 <code>stdout</code> 上显示一个字符串
SIn	0	1	从 <code>stdin</code> 读取一个字符串
S2B	1	1	转换为 <code>bool</code> : "false" or "f" or "0" = false "true" or "t" or "1" = true
S2C	1	1	转换为字符 <code>Char</code> （字符串中第一个字符的 ASCII 码）
S2I	1	1	转换为 <code>int</code> （如果字符串不是整数，则不执行任何操作）