RAM汇编语言编程

◆北京微芯力科技有限公司

苏兰冬

WinSilicon

数 据 处 理 指 令

- 简单的寄存器操作
- 立即数操作
- 寄存器移位操作
- 设置条码
- 条码的使用
- 乘法

WinSilicon

简单的寄存器操作

- 算术操作
- 按位逻辑操作
- 寄存器传送操作
- 比较操作

WinSilicon

简单的寄存器操作—算术操作

这类指令对两个32位操作数进行二进制算术操作（加、减和反向减，后者指把操作数次序颠倒后相减）。

```
ADD     ro, r1, r2          ;r0 = r1+r2
ADD     ro, r1, r2          ; r0 = r1+r2+C
SUB     ro, r1, r2          ; r0 = r1-r2
SBC     ro, r1, r2          ; r0 = r1-r2+C-1
RSB     ro, r1, r2          ; r0 = r2-r1
RSC     ro, r1, r2          ; r0 = r2-r1+C-1
```

WinSilicon

简单的寄存器操作—按位逻辑操作

这类指令对输入操作数的对应位进行指定的布尔逻辑操作。

```
ADD     r0,r1,r2        ;r0 = r1 and r2
ORR     r0,r1,r2        ;r0 = r1 or r2
EOR     r0,r1,r2        ;r0 = r1 xor r2
BIC     r0,r1,r2        ; r0 = r1 and not r2
```

简单的寄存器操作—寄存器传送操作

这些指令不用第一操作数，它在汇编语言格式中被省略。

```
MOV     r0,r2       ;r0 = r2
MVN     r0r,2       ;r0 = not r2
```

简单的寄存器操作—比较操作

这类指令不产生结果，仅根据所选择的操作来设置

```
CMP     r1,r2       ;根据r1-r2的结果设置cc
CMN     r1,r2       ;根据r1-r2的结果设置cc
TST     r1,r2       ;根据r1 and r2的结果设置cc
TEQ     r1,r2       ;根据r1 xor r2的结果设置cc
```

立即数操作

如果只希望把一个常数加到寄存器，而不是两个寄存器相加，可以用立即数值取代第二位操作数。
立即数通常是常量（literal)，前面加“#”

ADD　　r3,r3　#1;　　r3:=r3+1
AND　　r8,r7　# & ff;　　$r8:=r7_{[7:0]}$

立即数=（0 $\rightarrow$ 255）$X2^{2n}$

WinSilicon

寄存器移位操作

第三种定义数据操作的方式同第一种类似，但允许第二个寄存器操作数在同第一操作数运算之前完成移位操作，例:

```
ADD       r3, r2, r1,  LSL # 3          ;r3:=r2+8xr1
```

注意：它是一条RAM指令，在单个时钟周期内执行

设置条件码

如果程序员要，那么任何数据处理指令都能设置条件码（N、Z、C和V）。比较操作只能设置条件码。
要求以增加S操作码来指明，意为“设置条件码）例：

```
ADDS        r2, r2, r0          ;32位进位输出→C……
ADC         r3, r3, r1          ;…再加到高位字节中
```

条件码的使用

- 通过条件码转移指令来控制程序流

乘　法

有专门的数据处理指令支持乘法，即

MUL　　r4,r3,r3　　;r4:=(r3xr2)$_{[31:0]}$

它同其他算术指令有一些不同，即

- 不支持第二位操作数为立即数
- 结果寄存器不允许同为第一源寄存器
- 如果设置位S，则标志位V保留，而且标志位C不再有意义。

乘加指令

MLA　　r4,r3,r2,r1　　;r4:=(r3xr2+r1) $_{[31:0]}$

WinSilicon

数 据 传 送 指 令

- 寄存器间接寻址
- 初始化地址指针
- 单寄存器Load和Store指令
- 基址偏移寻址
- 多寄存器数据传送
- 堆栈寻址
- 块拷贝寻址

寄存器间接寻址

间接寄存器寻址利用一个寄存器的值（基址寄存器）作为存储器地址，或者从该地址取值存放到寄存器，或者将另一个寄存器的值存入该存储器地址。

指令汇编语言格式如下:

LED　　r0, [r1]　　;r0:=mem_{32}[r1]

STR　　r0, [r1]　　;mem_{32}[r1]:=r0

初始化地址指针

要访问一个特定的存储器单元，必须把一个RAM寄存器初始化，使之包含这个单元的地址。

作为一个例子来考虑，它必须从TABLE1向TABLE2拷贝数据， TABLE1和TABLE2都接近代码。

```
COPY    ADR    r1,TABLE1          ;r1指向TABLE1
        ADR    r2,TABLE2          ;r2指向TABLE2
        …
TABLE1  …                         ; 〈数据源〉
TABLE2  …                         ; 〈目标〉
```

WinSilicon

单寄存器Load和Store指令

这些指令使用基址寄存器来计算传送数据的地址。基址寄存器应该包含一个接近目标地址的地址，还要计算偏移量。

下列指令中没有偏移量

```
LDR     r0, [r1]        ;r0:=mem32[r1]
STR     r0, [r1]        ;mem32[r1]:=r0
```

现在把一个表中的第一个字拷贝到另一表中，即：

```
COPY    ADR     r1,TABLE1       ;r1指向TABLE1
        ADR     r2,TABLE2       ;r2指向TABLE2
        LDR     r0, [r1]        ;加载第一个数据…
        STR     r0, [r2]        ;将它存入 TABLE2
TABLE1  …                       ; 〈数据源〉
TABLE2  …                       ; 〈目标
```

基址偏移寻址

如果基址寄存器不包含确切的地址，则可以在基址上加上不超过4KB的[偏移量来计算传送地址，即：

LDR　　r0, [r1，#4]　　;r0:=mem_{32}[r1+4]

（此为前变址寻址模式）

LDR　　r0, [r1]，#4　　;r0:=mem_{32}[r1]

;r1:=r1+4

（此为后变址寻址模式）

多寄存器数据传送

当大量数据需要传送时，最好能同时存取几个寄存器，例：

LDMIA r1,{r0, r2, r5} ;r0:=mem_{32}[r1]

;r2:=mem_{32}[r1+4]

;r5:= mem_{32}[r1+8]

堆栈寻址

寻址变形基于的事实 → 在存储器中实现堆栈

RAM多寄存器传送指令支持全部4种形式的堆栈：

- 满递增
- 空递增
- 满递减
- 空递减

WinSilicon

块拷贝寻址

下面两条指令说明了这类指令的用途：

```
LDMIA       r0！，{r2- r9}
STMIA       r1，{r2- r9}
```

控 制 流 指 令

- 转移指令
- 条件转移
- 条件执行
- 转移和链接指令
- 子程序返回指令
- 监控程序调用
- 跳转表

转移指令

将程序的执行从一个位置切换到另一个位置最常用的方法是使用转移指令，即：

```
        B       LABEL

        …

LABEL   …
```

条件转移

一种典型的循环控制指令可能如下：

```
        MOV     r0,#0           ;计数器初始化
LOOP    …
        ADD     r0,r0,#1        ;循环计数器加1
        CMP     r0,#10          ;与循环的限制比较
        BEN     LOOP            ;如果不相等，则返回
        …                       ;否则循环中止
```

条件执行

- RAM指令集有一条不寻常的特征，就是条件执行不仅应用于转移指令，也应用于所有的RAM指令集。
- 一条指令本来用于跳过其后的几条指令，但如果给予这些指令以相反的条件，则转移将被忽略。

转移和链接指令

该指令完全象转移指令一样地执行转移，还把转移后面紧接的一条指令的地址保存到链接寄存器中。

```
        BL      SUBR        ; 转移到SUBR
        …                   ;返回到这里
SUBR    …                   ;子程序入口
        MOV     pc ,r14     ;返回
```

子程序返回指令

为了返回调用程序，必须将转移链接指令保存到r14中的值拷贝回程序寄存器

```
SUB2        … MOV     pc ,r14        ;把r14拷贝到r15来返回
```

WinSilicon

监控程序调用

- 只要程序需要输入或输出，例如把一些文本送到显示器，通常用监控程序，监控程序是一个运行于特权级别的程序。
- 提供了委托访问系统资源的方式
- 监控程序调用是在系统软件中实现的

跳转表

跳转表的思想是程序员有时想调用一系列子程序中的一个，而究竟调用哪一个须由程序员的计算而定。