AVR-MIPS-RISC-V 指令對照筆記

目錄

簡介

本文檔旨在提供 AVR、MIPS 和 RISC-V 三種處理器架構的指令集對照參考。這三種架構在嵌入式系統、學術研究和計算機架構教學中都非常重要。通過比較它們的指令，可以更好地理解不同架構的設計理念和特點。

- **AVR**：Atmel 公司開發的 8 位元 RISC 微控制器架構，廣泛應用於嵌入式系統
- **MIPS**：一種經典的 32/64 位元 RISC 架構，常用於學術教學和研究
- **RISC-V**：開源的指令集架構，模組化設計，支援 32/64/128 位元實現

指令集架構概述

特性	AVR	MIPS	RISC-V
位元數	8 位元	32/64 位元	32/64/128 位元
設計	RISC	RISC	RISC
指令長度	16/32 位元	32 位元	16/32/48/64 位元
通用寄存器數量	32 個 8 位元	32 個	32 個 (基本整數 ISA)
條件碼	有 (SREG)	無	無
開源性	否	否	是
延遲槽	無	有	無

基本指令對照表

數據傳輸指令

操作	AVR	MIPS	RISC-V
加載立即數	`LDI Rd, K`	`LI rt, imm` (偽指令)	`LI rd, imm` (偽指令)
從記憶體加載	`LD Rd, X/Y/Z`	`LW rt, offset(rs)`	`LW rd, offset(rs1)`
存入記憶體	`ST X/Y/Z, Rr`	`SW rt, offset(rs)`	`SW rs2, offset(rs1)`
寄存器間傳輸	`MOV Rd, Rr`	`MOVE rd, rs` (偽指令)	`MV rd, rs` (偽指令)
加載位元組	`LDS Rd, k`	`LB rt, offset(rs)`	`LB rd, offset(rs1)`
存入位元組	`STS k, Rr`	`SB rt, offset(rs)`	`SB rs2, offset(rs1)`

算術運算指令

操作	AVR	MIPS	RISC-V
加法	`ADD Rd, Rr`	`ADD rd, rs, rt`	`ADD rd, rs1, rs2`
加立即數	`SUBI Rd, K`	`ADDI rt, rs, imm`	`ADDI rd, rs1, imm`
帶進位加法	`ADC Rd, Rr`	-	-
減法	`SUB Rd, Rr`	`SUB rd, rs, rt`	`SUB rd, rs1, rs2`
帶借位減法	`SBC Rd, Rr`	-	-
乘法	`MUL Rd, Rr`	`MULT rs, rt`	`MUL rd, rs1, rs2`
除法	-	`DIV rs, rt`	`DIV rd, rs1, rs2`
比較	`CP Rd, Rr`	`SLT rd, rs, rt`	`SLT rd, rs1, rs2`

邏輯運算指令

操作	AVR	MIPS	RISC-V
與運算	`AND Rd, Rr`	`AND rd, rs, rt`	`AND rd, rs1, rs2`
或運算	`OR Rd, Rr`	`OR rd, rs, rt`	`OR rd, rs1, rs2`
異或運算	`EOR Rd, Rr`	`XOR rd, rs, rt`	`XOR rd, rs1, rs2`
左移	`LSL Rd`	`SLL rd, rt, sa`	`SLL rd, rs1, rs2`
右移	`LSR Rd`	`SRL rd, rt, sa`	`SRL rd, rs1, rs2`
算術右移	`ASR Rd`	`SRA rd, rt, sa`	`SRA rd, rs1, rs2`
位清零	`CLR Rd`	-	-
位測試	`SBRC Rr, b`	-	-

分支與跳轉指令

操作	AVR	MIPS	RISC-V
無條件跳轉	`JMP k`	`J target`	`JAL zero, offset`
子程序呼叫	`CALL k`	`JAL target`	`JAL ra, offset`
子程序返回	`RET`	`JR $ra`	`JALR zero, ra, 0`
等於時分支	`BREQ k`	`BEQ rs, rt, offset`	`BEQ rs1, rs2, offset`
不等於時分支	`BRNE k`	`BNE rs, rt, offset`	`BNE rs1, rs2, offset`
小於時分支	`BRLT k`	`BLT rs, rt, offset` (偽指令)	`BLT rs1, rs2, offset`
大於等於時分支	`BRGE k`	`BGE rs, rt, offset` (偽指令)	`BGE rs1, rs2, offset`

堆疊操作指令

操作	AVR	MIPS	RISC-V
壓入堆疊	`PUSH Rr`	`ADDI $sp, $sp, -4; SW rt, 0($sp)`	`ADDI sp, sp, -4; SW rs, 0(sp)`
彈出堆疊	`POP Rd`	`LW rt, 0($sp); ADDI $sp, $sp, 4`	`LW rd, 0(sp); ADDI sp, sp, 4`

系統與特殊指令

操作	AVR	MIPS	RISC-V
無操作	`NOP`	`NOP`	`ADDI x0, x0, 0` (等效NOP)
中斷啟用	`SEI`	`MFC0/MTC0`	`CSRRS`
中斷禁用	`CLI`	`MFC0/MTC0`	`CSRRC`
系統呼叫	-	`SYSCALL`	`ECALL`
斷點	`BREAK`	`BREAK`	`EBREAK`

尋址模式比較

尋址模式	AVR	MIPS	RISC-V
立即尋址	`LDI R16, 100`	`ADDI $t0, $zero, 100`	`ADDI x10, x0, 100`
寄存器尋址	`MOV R16, R17`	`ADD $t0, $t1, $zero`	`MV x10, x11`
直接尋址	`LDS R16, 0x100`	`LW $t0, 0x100`	`LW x10, 0x100`
間接尋址	`LD R16, Z`	`LW $t0, 0($t1)`	`LW x10, 0(x11)`
變址尋址	`LDD R16, Y+3`	`LW $t0, 3($t1)`	`LW x10, 3(x11)`
相對尋址	`RJMP label`	`J label`	`JAL x0, label`

寄存器對照表

功能	AVR	MIPS	RISC-V
零寄存器	-	`$zero`	`x0`
返回值	`R24:R25`	`$v0-$v1`	`x10-x11`
參數傳遞	`R8-R25`	`$a0-$a3`	`x10-x17`
臨時寄存器	`R16-R23`	`$t0-$t9`	`x5-x7, x28-x31`
保存寄存器	`R2-R15`	`$s0-$s7`	`x8-x9, x18-x27`
堆疊指標	`Y(R28:R29)`	`$sp`	`x2`
返回地址	-	`$ra`	`x1`
狀態寄存器	`SREG`	-	-

程式示例比較

簡單加法運算

AVR:

```asm
; 將兩個數字相加
LDI R16, 10    ; 第一個操作數
LDI R17, 20    ; 第二個操作數
ADD R16, R17   ; R16 = R16 + R17
```

MIPS:

```asm
# 將兩個數字相加
LI $t0, 10     # 第一個操作數
LI $t1, 20     # 第二個操作數
ADD $t2, $t0, $t1  # $t2 = $t0 + $t1
```

RISC-V:

```asm
# 將兩個數字相加
LI x10, 10     # 第一個操作數
LI x11, 20     # 第二個操作數
ADD x12, x10, x11  # x12 = x10 + x11
```

迴圈計數範例

AVR:

```asm
; 從10數到0
    LDI R16, 10         ; 初始計數值
loop:
    DEC R16             ; 計數器-1
    BRNE loop           ; 如果不是0，繼續迴圈
```

MIPS:

```asm
# 從10數到0
    LI $t0, 10          # 初始計數值
loop:
    ADDI $t0, $t0, -1   # 計數器-1
    BNE $t0, $zero, loop # 如果不是0，繼續迴圈
```

RISC-V:

```asm
# 從10數到0
    LI x10, 10          # 初始計數值
loop:
    ADDI x10, x10, -1   # 計數器-1
    BNE x10, x0, loop   # 如果不是0，繼續迴圈
```

子程序呼叫範例

AVR:

```asm
main:
    ; 主程序
    CALL subroutine     ; 呼叫子程序
    RJMP done           ; 跳到結束

subroutine:
    ; 子程序操作
    RET                 ; 返回

done:
    ; 程序結束
```

MIPS:

```asm
main:
    # 主程序
    JAL subroutine      # 呼叫子程序
    J done              # 跳到結束

subroutine:
    # 子程序操作
    JR $ra              # 返回

done:
    # 程序結束
```

RISC-V:

```asm
main:
    # 主程序
    JAL ra, subroutine  # 呼叫子程序
    JAL zero, done      # 跳到結束

subroutine:
    # 子程序操作
    JALR zero, ra, 0    # 返回

done:
    # 程序結束
```

參考資料

1. Atmel AVR 指令集手冊
2. MIPS 架構與組合語言程式設計
3. RISC-V 指令集手冊
4. Patterson, D. A., & Hennessy, J. L. (2017). 計算機組織與設計：硬體/軟體介面
5. Harris, S. L., & Harris, D. M. (2015). 數位設計和計算機架構