知识小料

「电计 2203 班」周常规知识整理共享

第 45 期

日期：2024-12-9

学科：计算机组成原理・实验

《计组实验一小记》

1 图与 PPT 部分

五条指令

1. `add.w  rd rj rk`
2. `addi.w  rd rj si12`
3. `ld.w  rd rj si12`
4. `st.w  rd rj si12`
5. `bne  rj rd offs16`

寄存器堆

- RA1, RA2, WA1 表示地址线。
- RD1, RD2, WD1 表示数据线。
- 在图中，RA1 连 rj，RD1 连 src1，表示在进入 ALU 前把 GR[rj] 的值作为 src1 加数传入。

注：`addi.w`和 `ld.w`不需要使用寄存器堆的 RA2。

控制信号表 控制信号表如下所示，注意 PPT 图中表示和代码表示的对应关系。

几个缩写：rf 的全称是 register file （寄存器堆），src 的全称是 source（源头），sel 的全称是 selection（选择）。

图示	代码表示	表示含义	牵扯指令
sel_alu_src2	src2_is_imm	**抉择**：操作数 2 是来自 RD2 的数，或者立即数（的符号扩展）	addi.w，ld.w，st.w（立即数）
sel_rf_res	res_from_mem	**抉择**：ALU 和是直接作为结果，或作为存储器的地址	ld.w（地址）
sel_rf_ra2	src_reg_is_rd	**抉择**：寄存器的 RA2 应准许 rk ，或者 rd	st.w，bne（准许 rd）
data_ram_we	mem_we	对存储器写使能信号	st.w
data_ram_ce	-	对存储器写片选控制信号	ld.w，st.w
br_taken	br_taken	**抉择**：PC 的目标采取 PC+4，或 br_target	bne（采取后者）
rf_we	gr_we	对寄存器堆的写使能信号（表示能否写寄存器堆）	add.w，addi.w，ld.w（开启）

表 1: 控制信号表

2 代码部分

顶层设计

- inst_sram_wen等 4 个：从指令存储器与 CPU 交互的 4 个信号（写使能、地址、写数据、读数据）。
- data_sram_wen等 4 个：从数据存储器与 CPU 交互的 4 个信号（写使能、地址、写数据、读数据）。
- clk：时钟。
- resetn：当取 1 时，reset取 0，valid取 1，pc取 nextpc。

指令分割 对 inst 进行分割。

- op_31_26 等 4 个：表示指令特定段的内容。
- rd：永远是 $\{4:0\}$。
- rj：永远是 $\{9:5\}$。
- rk：永远是 $\{14:10\}$。
- i12：12 位立即数，在 addi.w，ld.w，st.w 指令使用。
- i16：16 位立即数，在 bne 指令使用。

指令操作码译码 封装在黑箱子里。

```
wire [ 5:0] op_31_26;
wire [ 3:0] op_25_22;
wire [ 1:0] op_21_20;
wire [ 4:0] op_19_15;
```

随后是指令操作码的确定，需参考指令手册。

指令	31:26	25:22	21:20	19:15
add.w	000000	0000	01	00000
addi.w	000000	1010	-	-
ld.w	001010	0010	-	-
st.w	001010	0110	-	-
bne	010111	-	-	-

表 2: 指令操作码

控制信号 通过表 1 的内容，由指令种类控制各种信号。下面的列表中，「生效」表示该变量取 1，「不生效」表示该变量取 0。

- src2_is_imm ：对指令 addi.w，ld.w，st.w 生效，表示寄存器第二操作数 RD2 为立即数。
- res_from_mem ：对指令 ld.w 生效，表示 ALU 相加和作为存储器的地址。
- gr_we ：对指令 add.w，addi.w，ld.w 生效，表示允许寄存器写数据。
- mem_we ：对指令 st.w 生效，表示允许存储器写数据。
- src_reg_is_rd：对指令 st.w，bne 生效，表示寄存器的第二地址 RA2 准许 rd。
- rf_raddr1：表示寄存器第一地址 RA1 的取值，默认为 rj。
- rf_raddr2：表示寄存器第二地址 RA2 的取值，当 src_reg_is_rd 生效时取 rd，不生效取 rk。

寄存器黑箱

- clk，raddr1，raddr2，rdata2，we，waddr，wdata 表示寄存器内外连接的引脚。

- 要填入的信息如凸出部分所示，比如 .waddr(rd) 意思是将 rd送入寄存器写地址的引脚，毕竟当寄存器需要被写入时(由指令 add.w，addi.w，ld.w控制着)，其地址仅可能是 rd。

```
regfile u_regfile(
    .clk    (clk       ),
    .raddr1 (  rf_raddr1  ),
    .rdata1 (rj_value),
    .raddr2 (  rf_raddr2  ),
    .rdata2 (rkd_value),
    .we     (gr_we     ),
    .waddr  (  rd         ),
    .wdata  (rf_wdata )
);
```

其他信息

- br_offs：对 bne 指令生效，用于确定 pc 跳转的偏移量。对照指令，其目标为 16 位立即数作符号扩展得到的数值。扩展方法：往后填两个 0；往前对其最高位重复填写 14 位。

  所以是 {{14{inst[25]}}, inst[25:10], 2'b00}

  或者是 {{14{i16[15]}}, i16[15:0], 2'b00}

- br_target：对 bne 指令生效，用于确定 pc 跳转的目标。
- rj_eq_rd：对 bne 指令生效，表示读取 rj_value 和 rkd_value 是否相等。其中 rj_value 和 rkd_value 在寄存器黑箱里面通过 raddr1 和 raddr2 读出。
- br_taken：对 bne 指令生效，意为控制 pc 使用 br_target（而非 pc+4）的条件。需要满足以下三点：

  1. valid取 1，也就是至少要开机；
  2. inst_bne取 1，也就是要满足 bne指令的操作码（或者说指令的种类是 bne)；
  3. rj_eq_rd取 1，也就是指令所述的跳转条件。

- nextpc：对 bne 指令生效，确定下一个 pc 值。

- imm：对 addi.w，ld.w，st.w 指令生效，用于把 12 位立即数扩到 32 位。扩展方法：不往后填，往前对其最高位重复填写 20 位。
- alu_src1：对前四条指令生效，表示 ALU 的第一操作数，其值总是 GR[rj] 即 rj_value。
- alu_src2：对前四条指令生效，表示 ALU 的第二操作数。当 sel_alu_src2（代码表示为 src2_is_imm）生效时，其值为刚刚扩展过的立即数 imm，不生效时为 rkd_value。
- alu_result：对前四条指令生效，表示 ALU 加法的和。
- data_sram_we 等 3 个：对 ld.w，st.w 指令生效，是顶层设计中 CPU 和数据存储器交互的接口。
  1. data_sram_we 应取图上的 data_ram_we（代码表示为 ram_we）。
  2. data_sram_addr 应取图上的 ALU 加法结果，也就是代码的 alu_result。
  3. data_sram_wdata 对 st.w 指令生效，应取图上的 GR[rd] 也就是代码的 rkd_value。
- rf_wdata：表示对寄存器堆写入的数据，在图上的 sel_rf_res（代码表示 res_from_mem）生效时，取数据存储器读出的值 data_sram_rdata（一个顶层设计变量）；不生效时取 ALU 加法结果。