# 一、课程设计步骤

## 1、确定模型计算机功能及用途

完成一个较简单的计算机主机系统的设计，加深对微程序控制的计算机主机的基本构成，部件设计，部件间的连接，微程序的编制与调试等全过程的体验和认识。所设计的模型机应具备：在自行设计的模型机指令集基础上，运行简单用户程序，通过微程序实现对相应硬件的控制，实现模型机的特定功能，具体来说，应支持以下功能的用户程序实现：

1. 从内存中取两个数，相加后将结果存入第三个内存单元；
2. 在(1)的基础上，将相加过程中产生的“产生进位、溢出、结果为负、结果为0”状态存入状态寄存器（PSW）中;
3. 在以上的基础上，支持条件跳转功能，即“若状态寄存器具有某状态，则进行跳转”；
4. 实现乘法功能，采用累加方法，乘积16位；
5. 程序运行结束后停机。

为支持以上程序的运行，需要在控存中编写相应的微程序控制硬件，以支持指令集的功能。

## 2、指令系统

为实现相应功能，设计的指令系统有如下指令：

1. 取数指令LD Ri,AD，双操作数，微程序入口地址10H：

功能：将RAM中地址为AD的单元中数据存入寄存器Ri中，(AD)->Ri；

格式：双字长、双操作数。

第一个字节高四位为操作码1H，低两位为寄存器编号（00、01、10、11）。

第二个字节为AD，即数据来源地址（直接寻址）。

1. 存数指令ST Ri,AD，微程序入口地址20H：

功能：将Ri寄存器的内容存入RAM中地址为AD的内存单元，Ri->(AD)；

格式：双字长、双操作数。

第一个字节高四位为操作码2H，低两位为寄存器编号。

第二个字节为AD，即数据目标地址（直接寻址）。

1. 停机指令HALT，微程序入口地址30H：

功能：停机，此指令执行完毕后不再执行任何周期；

格式：单字长、无操作数，高四位为操作码3H。

1. 条件跳转指令JX A，微程序入口地址40H：

功能：分为JC、JV、JN、JZ，即“进位/溢出/为负/为零时跳转”，跳转意即将程序计数器PC的值加上A，即相对寻址。若条件不满足则PC+1->PC；

格式：双字长、双操作数。

第一个字节高四位为操作码4H，此后两位代表X：（00:C、01:V、10:N、11:Z）。

第二个字节为偏移量，用补码表示（立即数）。

1. 加法指令ADD Ri,Rj，微程序入口地址50H：

功能：将寄存器Ri、Rj的和存入寄存器Rj中，并将产生的状态打入PSW，(Ri)+(Rj)->Rj；

格式：单字长、双操作数。高四位为操作码5H，接下来两位代表i、最低两位代表j；

1. 无条件跳转指令JMP A，微程序入口地址60H：

功能：将PC的值加上A，即相对寻址；

格式：双字长、单操作数。首字节高四位为操作码6H，第二个字节为偏移量（立即数）。

1. 递增指令INC Ri，微程序入口地址70H：

功能：将寄存器Ri的值加1模256（按无符号数处理）；

格式：单字长、单操作数，高四位为操作码7H，低两位代表i。

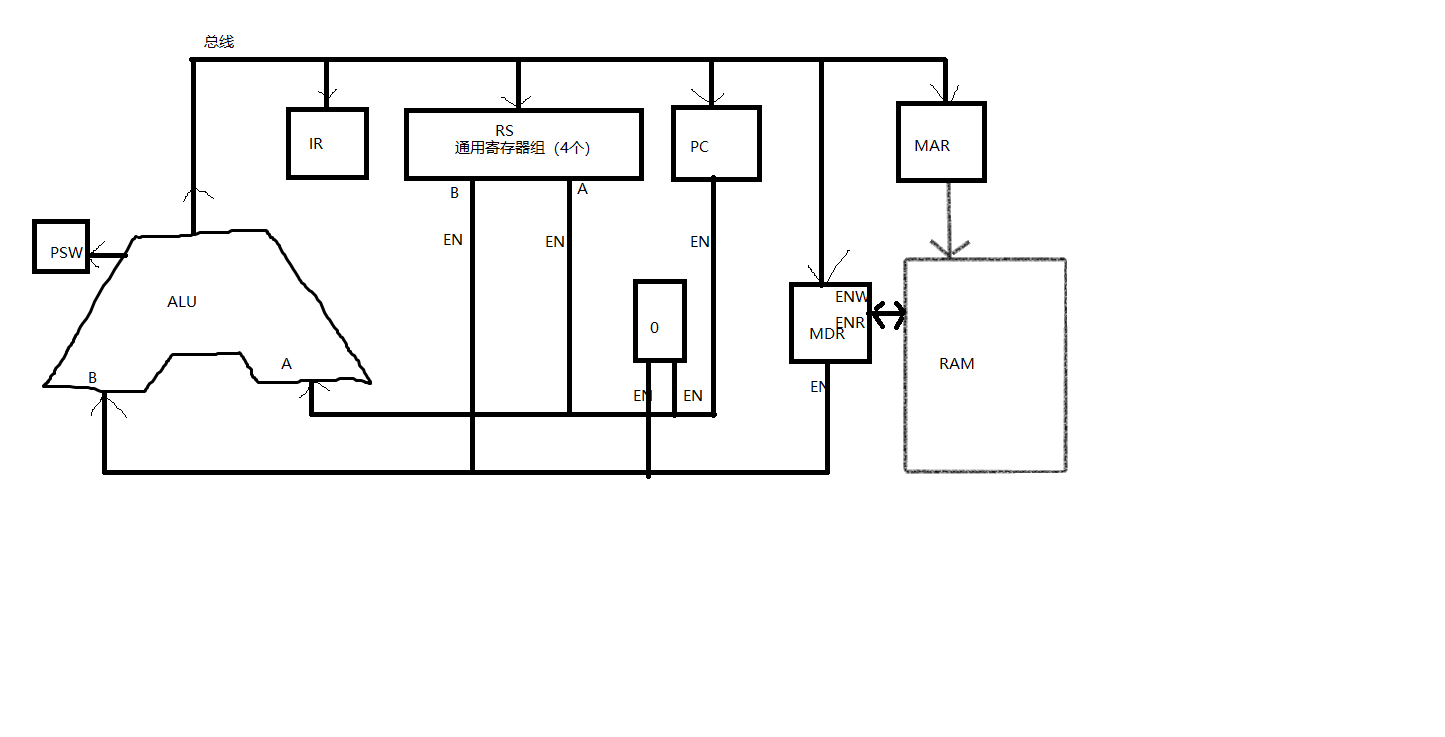
1. 带状态递增指令INC+ Ri，微程序入口地址A0H：

功能：同递增指令，但将在递增过程中产生的状态打入PSW；

格式：单字长、单操作数，高四位为操作码AH，低两位代表i。

## 3、总体结构与数据通路

总体结构图：



IR输出的与控存（ROM）中输出的控制信号、uPC、时钟信号等结构未画出。

该模型机的数据通路是以总线为基础，以ALU为核心构成的，寄存器间数据传输均需经过ALU，包括直送。每条数据传输指令执行时，数据通常由某寄存器使能输出到ALU参与运算（可能是直送）后，根据脉冲打入到对应寄存器中。

打入脉冲共6种。他们可以将总线上的数据打入到相应的寄存器。

分别为：CPPSW、CPIR、CPRS、CPPC、CPMDR、CPMAR。

其中CPPSW独立于其他五种，另外五种由单一控制信号决定，故互斥。

三态门输出使能信号共6种（扩展要求需要额外的一种，见扩展说明部分）。他们控制ALU的A、B两端的数据来源。

与A相连的有三种：ENRS(A)、EN0、ENPC，即A端数据来源；

与B相连的有三种：ENRS(B)、EN0、ENMDR，即B端数据来源。

此外还有MDR与RAM交互的双向三态门，有ENwrite与ENread两种使能信号，控制RAM的读写，实现了CPU与内存数据的隔离，形成一条单独的数据通路。

## 4、指令执行流程

所有指令按地址顺序，从00单元起存放于RAM中，由程序计数器控制访问，从而执行每一条指令。除相对跳转的指令JX与JMP外，指令按地址顺序执行（每次访问指令所在单元后PC+1->PC）。

由于寻址方式仅有直接寻址与立即数，且指令条数较少，每条指令的执行仅有两个周期：

1. 取指周期。此周期内流程由控存内入口地址为00的取指微程序控制（见附录2），该周期开始时PC指向该指令的地址，微程序控制数据流向如下（具体控制方式见下一节）：

PC->MAR; (PC+1->PC); RAM->MDR->IR;(QJP)

从而实现将RAM中该指令首字节的内容写入IR内，再按IR内操作码转移，将uPC转移到该指令对应的微程序入口，取指周期完成。

1. 执行周期。控存开始执行微程序，IR的低四位可能有参数（通用寄存器编号、JX指令的CVNZ），在微程序执行时起控制作用。微程序的末尾通过JP(0)指令，uPC转移到00地址，从而进入下一条指令的取指周期。

如此按规定的顺序执行每个指令周期，直到执停机指令的执行周期后，模型机不再工作。

以简单加法（附录表3-2）为例，示意指令执行流程，其中涉及的微操作具体实现见下一节：

取指周期

执行周期

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| oo |  |  |  |  |
| ↓ |  |  |  |  |
| PC->MAR |  |  |  |  |
| ↓ |  |  |  |  |
| PC+1->PC |  |  |  |  |
| ↓ |  |  |  |  |
| RAM->MDR |  |  |  |  |
| ↓ |  |  |  |  |
| MDR->IR |  |  |  |  |
| ↓ |  |  |  |  |
| QJP：按操作码转移 |  |  |  |  |
| 10↓LD R0,09 | 10↓LD R1,0A | 50↓ADD R1.R0 | 20↓ST R0,0B | 30↓HALT |
| PC->MAR | PC->MAR | R1+R0->R0 | PC->MAR | HALT |
| ↓ | ↓ | ↓ | ↓ |  |
| PC+1->PC | PC+1->PC | JP | PC+1->PC |  |
| ↓ | ↓ |  | ↓ |  |
| RAM->MDR | RAM->MDR |  | RAM->MDR |  |
| ↓ | ↓ |  | ↓ |  |
| MDR->MAR | MDR->MAR |  | MDR->MAR |  |
| ↓ | ↓ |  | ↓ |  |
| RAM->MDR | RAM->MDR |  | Ri->MDR |  |
| ↓ | ↓ |  | ↓ |  |
| MDR->R0 | MDR->R1 |  | MDR->RAM |  |
| ↓ | ↓ |  | ↓ |  |
| JP | JP |  | JP |  |

（手绘时不画表格框，括号内删除）

说明： 1、LD、ST指令均采用双字长、立即数寻址方式；

2、送指令地址的微操作安排在取指阶段的第一拍；

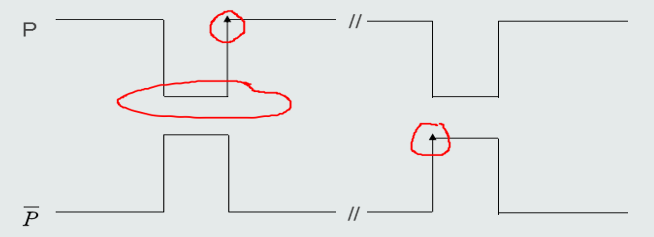
3、取指微程序入口固定位控存的00H，通过设置uPC的初始值为0实现；

4、第一条机器指令放在00H开始的RAM单元。通过设置MAR初值为0实现第一条机器指令的取指数。

## 5、微程序流程

模型机采用微程序控制模式，将完成某任务需要的控制命令组合在一起连续地写在控存中形成微程序，供指令通过入口调用。每个控制命令对应一种微操作，这需要一定的时序与设计可以完成。

1. 微程序控制器时序



（手绘时去掉红圈，括号内删除）

将!uRD接地，将控存设为始终可读。

P脉冲的上升沿将独处的微指令送往uIR。

!P脉冲将运算结果送往指定的寄存器，并将脉冲打入uPC形成下一条微指令的地址。

1. 微指令格式
   1. 微指令字段定义

a) ALU-B端数据来源：uIR15 uIR14

0 0 EN0

0 1 ENRS

1 0 ENMDR

1 1 ENXJP（见扩展要求）

b) ALU-A端数据来源：uIR13 uIR12

0 0 EN0

0 1 ENRS

1 0 备用

1 1 ENPC

c) 输出分配（打入脉冲）：uIR11 uIR10 uIR9

0 0 0 CPRS

0 0 1 备用

0 1 0 备用

0 1 1 备用

1 0 0 CPIR

1 0 1 CPMDR

1 1 0 CPMAR

1 1 1 CPPC

d) 低位进位控制：uIR8

1. C0=0
2. C0=1

e) PSW打入脉冲：uIR6

1. 有CPPSW
2. 无CPPSW

f) 存储器读写控制：uIR5 uIR4

0 1 !WR

1 0 !RD

g) 停机控制：uIR3

0 不停机

1 停机

h) 后继微地址形成方式：

uIR2 uIR1 uIR0

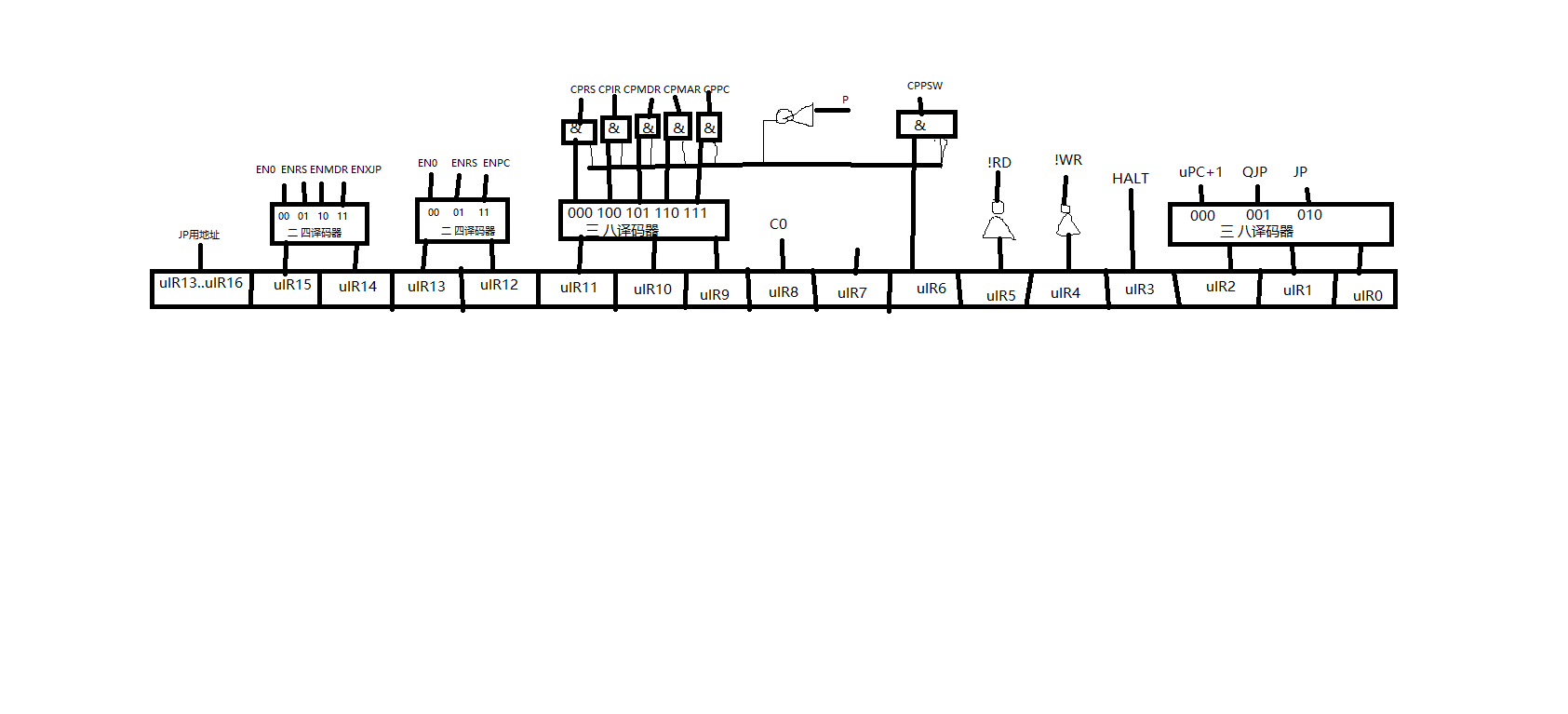
0 0 0 uPC+1，顺序执行

0 0 1 QJP，高四位按操作码转移，低四位为0

0 1 0 JP，无条件转移，微地址由uIR23~uIR16提供

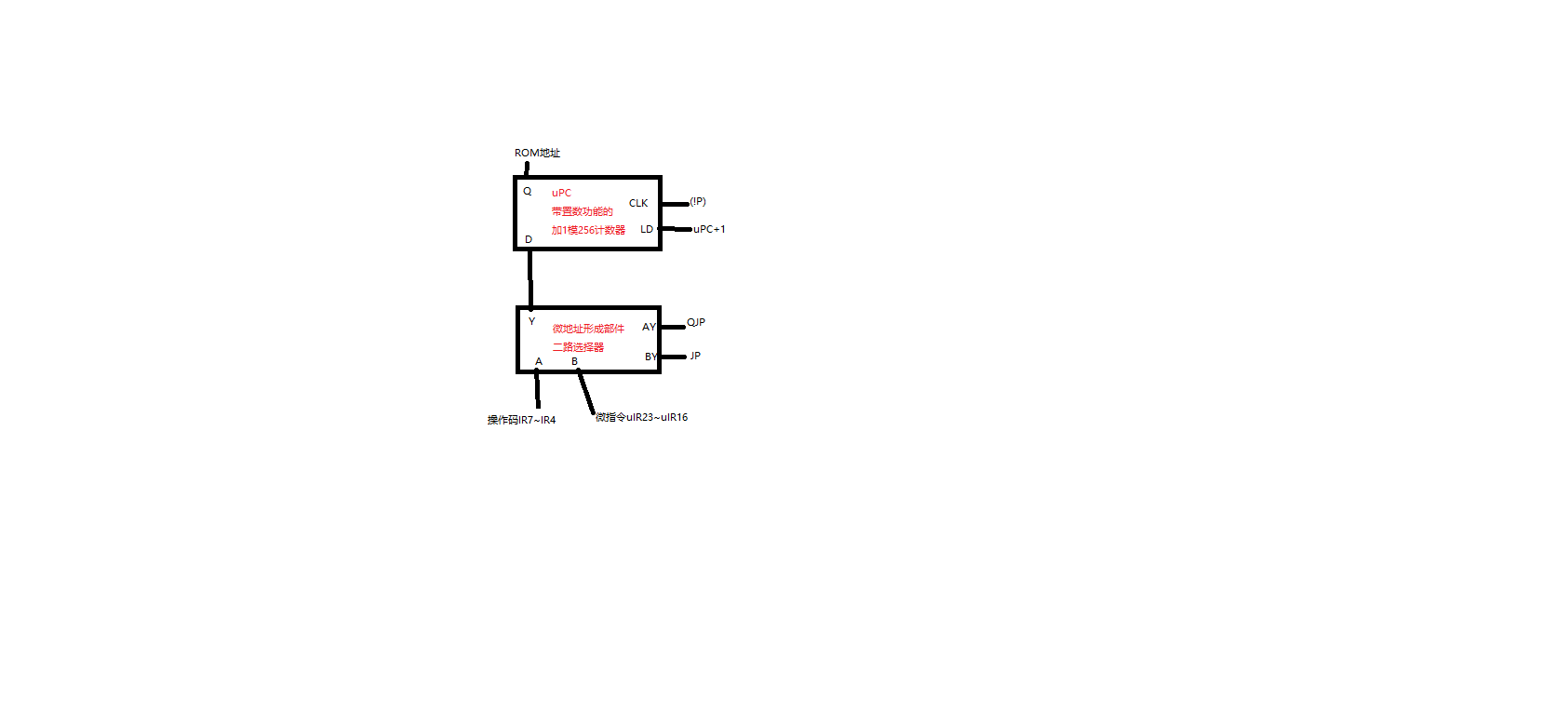
其他 备用

* 1. 微命令形成逻辑



* 1. 后继微地址产生逻辑

使用以下结构：



此处二路选择器经过特殊设计，即A端仅接受高四位，低四位始终为0。

由图可知：

* 1. uPC+1=1时，uPC+1
  2. QJP=1时，按操作码高四位转移
  3. JP=1时，按微指令无条件转移

（此三个信号由译码器产生，故互斥）

意即uPC+1=1时，Q=Q+1;

QJP=1时，Q=IR7IR6IR5IR40000；

JP=1时，Q=uIR23~uIR16

1. 支持的微操作

经过如上设计，下面列举单条微指令可以控制的信息流向，指令具体内容见附录2：

* 1. 通过总线

送地址指令PC->MAR：

PC->ALU-A->ALU->Bus->MAR

程序计数器加1 PC+1->PC：

PC->ALU-A->ALU+1->Bus->PC

取指令MDR->IR：

MDR->ALU-B->ALU->Bus->IR

取数到通用寄存器指令MDR->Ri：

MDR->ALU-B->ALU->Bus->RS

寻址指令MDR->MAR：

MDR->ALU-B->ALU->Bus->MAR

准备存数指令Ri->MDR：

RS->ALU-A->ALU->Bus->MDR

累加指令Ri+Rj->Rj：

RS->ALU-A→ALU(状态打入PSW)->Bus->RS

RS->ALU-B↗

无条件跳转指令MDR+PC->PC

MDR->ALU-B→ALU->Bus->PC

PC->ALU-A ↗

递增指令Ri+1->Ri

RS->ALU-A->ALU+1->Bus->RS

保存状态递增Ri+1->Ri+

RS->ALU-A->ALU+1(状态打入PSW)->Bus->RS

注：1、涉及寄存器组的指令，具体寄存器编号确认见通用寄存器组部件设计；

2、仅写明ALU一端数据来源的指令，另一端数据为0；

3、ALU+1代表C0为1；

4、条件跳转指令XJP+PC->PC见扩展部分的实现说明。

② RAM与MDR之间

读内存指令RAM->MDR：

RAM->MDR, RD=1

写内存执行MDR->RAM：

MDR->RAM, WR=1

1. 跳转指令

无条件转移指令JP

按操作码转移指令QJP

1. 停机指令

停机指令HALT

## 6、各部件设计

(1) 启停器

电路图见附录图1-23。

输出的脉冲前的3路与门分别连接IMPULSE脉冲与两个启动条件：

1. 停机控制信号HALT为低电平（无效）；
2. 启动信号START曾被置为高电平，使DFF触发器前的高电平被输出到与门。

清零信号CLR为低电平时，将DFF触发器输出置为低电平，等待下一次开机。

启动时，将HALT置为低电平、CLR置为高电平，然后触发START信号，即可使CP输出与连续脉冲频率相同的信号，作为微指令的时钟信号。

(2) 八位寄存器

电路图见附录图1-2。

由8个DFF触发器并联而成，可以通过CP上升沿统一置数、复位信号CLR统一复位。

模型机的MAR、IR不需要与运算器任何一端相连，故直接采用此种寄存器。其他寄存器中封装的均为此结构。

(3) 三态门

三态门使能端均为高电平有效。

① 单向三态门

见附录图1-16。

由8个三态门并联而成，使能信号EN可以统一对输入数据进行使能控制，扩展功能中条件跳转直接使用了该部件，该部件也是寄存器与ALU输入端相连的必要部件，否则会出现数据冲突的情况。

ALU的某端置零使用了接地线后接三态门接ALU输入端的结构，对该部件使能可使低电平输入到ALU，见附录图1-20。

② 双向三态门

见附录图1-18。

该元件由八个单位组成，每个单位由一个双向端口和两个方向不同的三态门组成，分别控制该端口的输入、输出，三态门另一侧连接相应的输入输出端口。由两个使能信号EN\_read、EN\_write控制八个双向端口的输入输出。

在RAM与MDR数据交互时有应用，故被封装入MDR中。

1. 内存数据寄存器MDR

见附录图1-24。

由三个单向三态门（A、B、C）、一个双向三态门D和一个八位寄存器封装而成。

使能信号EN\_write与EN\_read控制A、B、D的数据流向，从而控制MDR的数据来源：

1. EN\_write有效时，D控制MDR数据流向RAM。与RAM写信号!WR配合可以实现将MDR中数据写入RAM。
2. EN\_read有效时，A、B、D控制RAM数据流向MDR，同时切断Bus->MDR的数据通路（EN\_read无效时，Bus->MDR的数据通路总是开通）。与CP上升沿、读信号!RD配合，可以将RAM中数据打入MDR，实现数据的读出。

使能信号ENQ控制C的数据流向，控制MDR内数据流向ALU-B端。