一、**023A内核的大致结构**

023A为四级单流水线，六大模块结构。

任何一条指令都可大致分为4步来完成，每一步就对应了流水线中的一个模块。

第一步：取指模块

根据PC寄存器指向位置读取指令

解析出指令的基本类型(023A指令集指令分32个大类，每个大类又可细分小类)

如果是跳转类型指令，就根据跳转指令的细分类型/信息修改PC寄存器，完成跳转。

在后续流水线层级中，跳转类型指令直接等同与空指令，不进行任何处理。

第二步：解析模块

解析出指令的细分类型，以及指令的源操作数、目标操作数寄存器号的信息。

并根据源操作数寄存器号，从寄存器中取出数据。

将这些信息全部传递给下一级流水线。

第三步：处理模块

根据指令的大类型，判断使用哪个功能模块执行对应功能。将细分类型信息和源操 作数信息传递给对应的功能模块。目标寄存器号的信息原封不动。

(可以分为2大类功能模块：内存读写模块 数据运算模块

其中数据运算模块又可分为：浮点运算模块 整型运算模块 位运算模块)

当功能模块处理完毕后，将运算结果和回写的目标寄存器号信息传递给下一级流水 线。

第四步：回写模块

如果有数据要回写到寄存器(例如写内存就没有回写的需求)，就根据上一级传来的 回写数据、寄存器号信息将数据写入到指定寄存器中。

数据旁路模块：

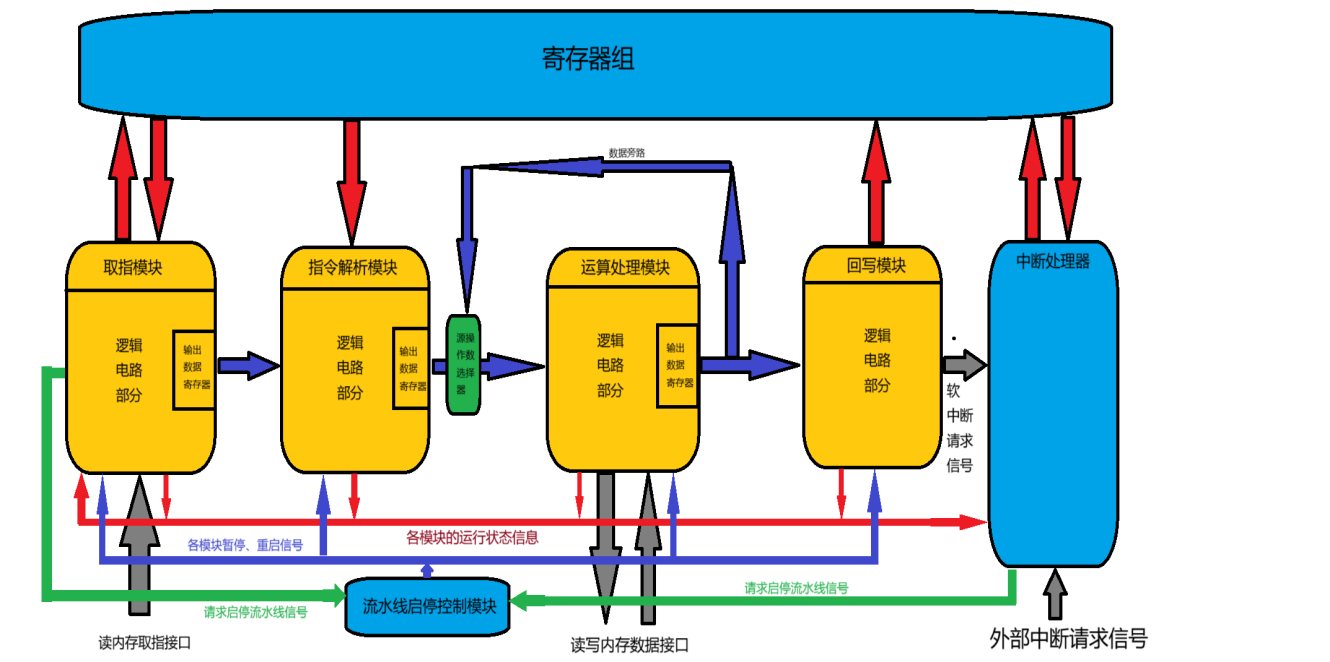
数据旁路模块不属于流水线中的任意层级。由回写模块的起始端反向连接到处理模 块起始端。其作用是解决因为数据冒险，被迫需要插入空指令的问题。

在处理模块接收源操作数数据时，如果源操作数的寄存器号与当前回写模块中准备 回写数据的寄存器号相同，就直接通过数据旁路通道读取回写数据替代掉解析模块 从寄存器中取出的操作数。

寄存器模块：

存储CPU运行过程中的寄存器数据。

中断处理模块：

负责处理中断信号。如果检测到中断信号。CPU将停止取指模块的工作，不再进行 进一步的取指。并开始监测流水线中所有模块，当发现这些模块已经将未完成的任 务全部完成后，便启动中断响

1. **中断处理的详细流程**
2. 根据中断号找到中断向量表中存储对应中断跳转地址的地址。

中断向量表固定占用内存 0-1023 的区间。其中每个中断跳转地址占用4个字节。

故总计256个中断号。

023A的中断号从0开始编码，故中断号可为0-255。

存储中断跳转地址的内存地址 = 中断号 \* 4

1. 暂停取指模块的工作，使流水线不再加载执行新的指令。并等待当前

流水线中所有未完成的指令执行完毕。

1. 将PC寄存器数据存入IPC，将取得的中断号跳转地址存入PC寄存器。

将SYS寄存器数据清0 (表示让CPU关闭保护模式、关闭虚拟内存、关闭中断响应)

1. 重新开启取指模块的工作

此时，CPU已经开始执行中断处理函数中的指令。

根据中断处理的详细流程，可以知道CPU进入中断处理后仅仅是修改了PC/IPC/SYS寄存器，

从而调整了CPU运行模式、保存中断返回地址、跳转入中断处理函数地址。

故，需要注意如果在中断处理函数中要使用任何寄存器，都应当先手动编写代码将对应的寄存器数据存入堆栈。

其次，在进入中断处理函数后，CPU默认是禁用了中断响应。如果后续要允许CPU可以响应中断，需要手动调用EI指令。且在此之前需要将IPC寄存器数据入栈保存，防止重复中断覆盖掉原中断返回地址的信息。

1. **CPU重启后寄存器的初始值**

普通数值寄存器: R1 R2 R3 R4 R5 R6 R7(CS) R8(DS) 全部=0

特殊功能寄存器: SP 栈顶寄存器 =0

PC 程序指针寄存器 =1024

TPC 普通函数调用链接寄存器 =0

IPC 中断处理函数链接寄存器 =0

FLAG 标志位寄存器 =0

SYS 处理器运行模式寄存器 =0

TLB 虚拟内存TLB表起始地址寄存器 =0

除PC寄存器外所有寄存器全部为0，PC寄存器的初始值是1024，也就是CPU重启后的程序入口地址。如果裸机开发的情况下，main函数的存储地址必须对齐1024，如此CPU启动后才能直接进入main函数中执行指令。

其中SP也是为0，意味着CPU启动后不存在堆栈空间。需要自行编写代码修改SP寄存器的地址，才能使用堆栈功能。例如MOV SP,65536D;将SP寄存器初始值改为65536内存处，堆栈也就以此地址为底向下生长。

1. **内存数据的大小端**

CPU内存数据的存储分为大端方式和小端方式。

大端存储方式:

CPU寄存器中的数据在存入内存时，高位字节存在低内存地址，低位字节存在高位地址。

示例: UINT32: 0x11223344 数据 存放到内存 0x80处

内存地址 80 81 82 83

字节数据 11 22 33 44

小端字节数

CPU寄存器中的数据在存入内存时，高位字节存在高内存地址，低位字节存在低位地址。

示例: UINT32: 0x11223344 数据 存放到内存 0x80处

内存地址 80 81 82 83

字节数据 44 33 22 11

具体是大端存储方式还是小端存储方式，取决于CPU的指令集设计。例如常见的

指令集中x86是小端方式，arm可通过指令自行调整大小端，mips是大端方式。

而023A内核CPU就是采用的大端方式。

字节序的不同，在部分场合下会直接影响程序的执行结果。所以有必要了解。

C语言中代码判断CPU大小端字节序的方法

#include <stdint.h>

uint8\_t getMemEndian(){

static union{

uint16\_t a;

uint8\_t b[2];

}test = {.a = 0x00ff};

return test.b[0];

}

如果是大端字节序，返回0xff；小端字节序返回0x00

1. **CPU内核对外开放的接口作用**

module OpenQinling\_023A(

input clk,//时钟接口

////////////////CPU中断接口//////////////////

input int\_ask,//中断请求接口

input[7:0]int\_num,//中断号输入接口[0-255]

////////////////CPU连接MMU的接口//////////////////

output is\_enable\_vm,//虚拟内存是否启用

output[31:0]tlb\_address,//虚拟内存tlb表地址输出[0-4GB]

/////////CPU加载指令的接口，连接内存//////////////////

input [31:0]order\_bus,//指令加载总线[4bytes]

output [31:0]order\_address,//指令地址输出[0-4GB]

input order\_read\_cplt,//指令内存响应信号输入

////////CPU读写内存数据的接口，连接内存与外设//////////

input [31:0]data\_bus\_r,//内存读总线

output[31:0]data\_bus\_w,//内存写总线

output [31:0]data\_address,//内存地址输出[0-4GB]

output [1:0]data\_rw,//内存读写模式控制[0/1不读不写,2读,3写]

output [1:0]data\_size,//单次内存读写字节数控制[1-4bytes]

input data\_rw\_cplt,//数据内存响应信号输入

//接收当读写的设备完成此次读写任务后发出的响应信号

//////////CPU状态信息输出的接口，连接内存与外设/////////////////

output stoping,//cpu暂停中(当出现信号，所有正在执行被CPU读写操作的

//外设哪怕完成了读写，也不能立刻结束此次读写任务，要保持输出

//读写完成响应信号data\_rw\_cplt)

output intering,//cpu响应中断中(当出现信号，所有正在执行被CPU读写操作的

//外设都要终止此次读写操作)

output rsting,//cpu重启中(当出现信号，所有外设必须全部跟随重启)

///////////cpu内部寄存器数据查看接口(用于Debug)///////////////

output[31:0]r1\_d,r2\_d,r3\_d,r4\_d,r5\_d,r6\_d,cs\_d,ds\_d,flag\_d,

pc\_d,tpc\_d,ipc\_d,sp\_d,tlb\_d,sys\_d

);

CPU内核中不包括MMU和缓存器，需要自己开发与挂接。

1. **指令结构**

023A指令集中指令全部都是32位的等长指令。

指令大致可分为8种结构。同种结构的指令，32个bit位表示的作用相同；

**结构类型1：无参数指令（即指令中不附带任何参数）**

**格式特点：**

[31:27]指令大类编号

[26:0] 指令细分类型编号

**该类指令有：**

空指令：大类编号0，无细分类型

NOP [31:27]=0 [26:0]=0

链接跳转型指令：大类编号14

RT [31:27]=14 [26:0]=0

IT [31:27]=14 [26:0]=1

EIT [31:27]=14 [26:0]=2

VIT [31:27]=14 [26:0]=3

PVT [31:27]=14 [26:0]=4

CPU模式控制类指令： 大类编号15

EI [31:27]=15 [26:0]=0

DI [31:27]=15 [26:0]=1

EP [31:27]=15 [26:0]=2

EV [31:27]=15 [26:0]=3

DV [31:27]=15 [26:0]=4

CPU重启指令：大类编号31，无细分类型

RST [31:27]=31 [26:0]=0

**结构类型2：格式转换运算、转移运算、比较运算**

**格式特点：**

[31:27]指令大类编号

[26:24]细分类型编号

[23:20]参数a要操作寄存器的编号

[19:16]参数b要操作寄存器的编号

[15:0]如果参数b的编号为0，那就是一个立即数型参数，立即数存储在此

(可存储0-65535的uint16型立即数)

**该类指令有：**

转移数据、初始化寄存器指令：大类编号9

MOV a,b [31:27]=9 [26:24]=0 [23:20]=a寄存器 [19:16]=b寄存器 [15:0]=b立即数

ASS a,b [31:27]=9 [26:24]=0 [23:20]=a寄存器 [19:16]无用位 [15:0]=b立即数

(ASS指令在汇编语言中基本不用，但INIT/INIT64指令都是由ASS和MOV指令所构 成的伪指令。其作用是 a=(a<<16)|b 参数b只可为立即数)

浮整数据类型转换、正负转换指令：大类编号4

UITF a,b [31:27]=4 [26:24]=0 [23:20]=a寄存器 [19:16]=b寄存器 [15:0]=b立即数

ITF a,b [31:27]=4 [26:24]=1 [23:20]=a寄存器 [19:16]=b寄存器 [15:0]=b立即数

FTI a,b [31:27]=4 [26:24]=2 [23:20]=a寄存器 [19:16]=b寄存器 [15:0]=b立即数

TPN a,b [31:27]=4 [26:24]=3 [23:20]=a寄存器 [19:16]=b寄存器 [15:0]=b立即数

无符号整型比较运算指令：大类编号20

UEC a,b [31:27]=20 [26:24]=0 [23:20]=a寄存器 [19:16]=b寄存器 [15:0]=b立即数

UNEC a,b [31:27]=20 [26:24]=1 [23:20]=a寄存器 [19:16]=b寄存器 [15:0]=b立即数

UMC a,b [31:27]=20 [26:24]=2 [23:20]=a寄存器 [19:16]=b寄存器 [15:0]=b立即数

ULC a,b [31:27]=20 [26:24]=3 [23:20]=a寄存器 [19:16]=b寄存器 [15:0]=b立即数

UMEC a,b [31:27]=20 [26:24]=4 [23:20]=a寄存器 [19:16]=b寄存器 [15:0]=b立即数

ULEC a,b [31:27]=20 [26:24]=5 [23:20]=a寄存器 [19:16]=b寄存器 [15:0]=b立即数

整型比较运算指令：大类编号21

EC a,b [31:27]=21 [26:24]=0 [23:20]=a寄存器 [19:16]=b寄存器 [15:0]=b立即数

NEC a,b [31:27]=21 [26:24]=1 [23:20]=a寄存器 [19:16]=b寄存器 [15:0]=b立即数

MC a,b [31:27]=21 [26:24]=2 [23:20]=a寄存器 [19:16]=b寄存器 [15:0]=b立即数

LC a,b [31:27]=21 [26:24]=3 [23:20]=a寄存器 [19:16]=b寄存器 [15:0]=b立即数

MEC a,b [31:27]=21 [26:24]=4 [23:20]=a寄存器 [19:16]=b寄存器 [15:0]=b立即数

LEC a,b [31:27]=21 [26:24]=5 [23:20]=a寄存器 [19:16]=b寄存器 [15:0]=b立即数

浮点型比较运算指令：大类编号22

FEC a,b [31:27]=22 [26:24]=0 [23:20]=a寄存器 [19:16]=b寄存器 [15:0]=b立即数

FNEC a,b [31:27]=22 [26:24]=1 [23:20]=a寄存器 [19:16]=b寄存器 [15:0]=b立即数

FMC a,b [31:27]=22 [26:24]=2 [23:20]=a寄存器 [19:16]=b寄存器 [15:0]=b立即数

FLC a,b [31:27]=22 [26:24]=3 [23:20]=a寄存器 [19:16]=b寄存器 [15:0]=b立即数

FMEC a,b [31:27]=22 [26:24]=4 [23:20]=a寄存器 [19:16]=b寄存器 [15:0]=b立即数

FLEC a,b [31:27]=22 [26:24]=5 [23:20]=a寄存器 [19:16]=b寄存器 [15:0]=b立即数

整型位数转换指令：大类编号18

BTD a,b [31:27]=18 [26:24]=0 [23:20]=a寄存器 [19:16]=b寄存器 [15:0]=b立即数

WTD a,b [31:27]=18 [26:24]=1 [23:20]=a寄存器 [19:16]=b寄存器 [15:0]=b立即数

**结构类型3：数据运算指令**

**格式特点：**

[31:27]指令大类编号

[26:24]细分类型编号

[23:20]参数b源操作数寄存器的编号

[19:16]参数c源操作数寄存器的编号

[15:12]参数a目标操作寄存器的编号

[11:0]如果参数c的编号为0，那就是一个立即数型参数，立即数存储在此

(可存储0-4095的uint12型立即数)

无符号整型四则运算指令：大类编号1

UADD a,b,c [31:27]=1 [26:24]=0 [23:20]=b寄存器 [19:16]=c寄存器

[15:12]=a目标寄存器 [11:0]=c立即数

USUB a,b,c [31:27]=1 [26:24]=1 [23:20]=b寄存器 [19:16]=c寄存器

[15:12]=a目标寄存器 [11:0]=c立即数

UMUL a,b,c [31:27]=1 [26:24]=2 [23:20]=b寄存器 [19:16]=c寄存器

[15:12]=a目标寄存器 [11:0]=c立即数

UDIV a,b,c [31:27]=1 [26:24]=3 [23:20]=b寄存器 [19:16]=c寄存器

[15:12]=a目标寄存器 [11:0]=c立即数

整型四则运算指令：大类编号2

ADD a,b,c [31:27]=2 [26:24]=0 [23:20]=b寄存器 [19:16]=c寄存器

[15:12]=a目标寄存器 [11:0]=c立即数

SUB a,b,c [31:27]=2 [26:24]=1 [23:20]=b寄存器 [19:16]=c寄存器

[15:12]=a目标寄存器 [11:0]=c立即数

MUL a,b,c [31:27]=2 [26:24]=2 [23:20]=b寄存器 [19:16]=c寄存器

[15:12]=a目标寄存器 [11:0]=c立即数

DIV a,b,c [31:27]=2 [26:24]=3 [23:20]=b寄存器 [19:16]=c寄存器

[15:12]=a目标寄存器 [11:0]=c立即数

浮点运算指令：大类编号3

FADD a,b,c [31:27]=3 [26:24]=0 [23:20]=b寄存器 [19:16]=c寄存器

[15:12]=a目标寄存器 [11:0]无用位

FSUB a,b,c [31:27]=3 [26:24]=1 [23:20]=b寄存器 [19:16]=c寄存器

[15:12]=a目标寄存器 [11:0]无用位

FMUL a,b,c [31:27]=3 [26:24]=2 [23:20]=b寄存器 [19:16]=c寄存器

[15:12]=a目标寄存器 [11:0]无用位

FDIV a,b,c [31:27]=3 [26:24]=3 [23:20]=b寄存器 [19:16]=c寄存器

[15:12]=a目标寄存器 [11:0]无用位

移位运算指令：大类编号5

SAL a,b,c [31:27]=5 [26:24]=0 [23:20]=b寄存器 [19:16]=c寄存器

[15:12]=a目标寄存器 [11:0]无用位

SAR a,b,c [31:27]=5 [26:24]=1 [23:20]=b寄存器 [19:16]=c寄存器

[15:12]=a目标寄存器 [11:0]无用位

SOL a,b,c [31:27]=5 [26:24]=2 [23:20]=b寄存器 [19:16]=c寄存器

[15:12]=a目标寄存器 [11:0]无用位

SOR a,b,c [31:27]=5 [26:24]=3 [23:20]=b寄存器 [19:16]=c寄存器

[15:12]=a目标寄存器 [11:0]无用位

比特、位运算指令：大类编号6

AND a,b,c [31:27]=5 [26:24]=0 [23:20]=b寄存器 [19:16]=c寄存器

[15:12]=a目标寄存器 [11:0]=c立即数

OR a,b,c [31:27]=5 [26:24]=1 [23:20]=b寄存器 [19:16]=c寄存器

[15:12]=a目标寄存器 [11:0]=c立即数

NOT a,c [31:27]=5 [26:24]=2 [23:20无用位 [19:16]=c寄存器

[15:12]=a目标寄存器 [11:0]=c立即数

XOR a,b,c [31:27]=5 [26:24]=3 [23:20]=b寄存器 [19:16]=c寄存器

[15:12]=a目标寄存器 [11:0]=c立即数

**结构类型4：跳转、软中断指令**

**格式特点：**

[31:27]指令大类编号

[26:0]立即数

JNE num [31:27]=10 [26:0]=跳转地址立即数

JMP num [31:27]=11 [26:0]=跳转地址立即数

JT num [31:27]=12 [26:0]=跳转地址立即数

INT num [31:27]=13 [26:8]无用位 [7:0]中断号立即数

**结构类型5：内存读写指令**

**格式特点：**

[31:27]指令大类编号

[26:26]读还是写

[25:24]读写的位宽

[23:20]读写数据存储的寄存器号

[19:16]读写地址的寄存器号

[15:0]读写地址立即数

RD 位宽,data,add [31:27]=7 [26:26]=0 [25:24]=位宽

[23:20]=data寄存器号 [19:16]=add寄存号

[15:0]=如果add=0,读地址的低16位立即数

WE 位宽,data,add [31:27]=7 [26:26]=0 [25:24]=位宽

[23:20]=data寄存器号 [19:16]=add寄存号

[15:0]=如果add=0,写地址的低16位立即数

**结构类型6：栈读写指令**

**格式特点：**

[31:27]指令大类编号

[26:25]读写的位宽

[24:21]读写数据存储的寄存器号

[20:0]读写地址的栈顶偏移立即数

SRD 位宽,data,offset [31:27]=16 [26:25]=位宽 [24:21]=data寄存器号

[20:0]=offset读栈顶偏移地址的立即数

SWE 位宽,data,offset [31:27]=16 [26:25]=位宽 [24:21]=data寄存器号

[20:0]=offset写栈顶偏移地址的立即数

**结构类型7：压栈出栈指令**

**格式特点：**

[31:27]指令大类编号

[26:26]压栈还是出栈

[25:24]读写的位宽

[23:20]无用位

[19:16]出入栈的寄存器号

[15:0]如果是入栈，可以是入栈一个立即数，存储立即数

POP 位宽,data [31:27]=8 [26:26]=0 [25:24]=位宽

[23:20]=无用位 [19:16]=data寄存器号

[15:0]=无用位

PUSH位宽,data [31:27]=8 [26:26]=1 [25:24]=位宽

[23:20]=无用位 [19:16]=data寄存器号

[15:0]=如果寄存器号=0,存储入栈的立即数

**结构类型8：比特位操作指令**

**格式特点：**

[31:27]指令大类编号

[26:24]子模式

[23:20]目标寄存器号

[19:16]源操作数寄存器号

[15:10]源操作数为立即数情况下，存储立即数(0-63)

[9:5]操作的高限位

[4:0]操作的低限位

SET 高限位-低限位,a,b [31:27]=19 [26:24]=0 [23:20]=a寄存器号

[19:16]=b寄存器号 [15:10]=b立即数

[9:5]=高限位 [4:0]=低限位

GET 高限位-低限位,a,b [31:27]=19 [26:24]=0 [23:20]=a寄存器号

[19:16]=b寄存器号 [15:10]=b立即数

[9:5]=高限位 [4:0]=低限位