0.概念

四大原则（简单规则越小越快 相互妥协 常用的快速运行）

1.汇编指令

指令类型及常用指令格式（加一下xor，or，and）（后面带i,u的东西）

syscall

printf- move $a0,$v0 | li $v0,1 |syscall

scanf- |li $v0, 5|syscall

地址跳转方式

<=的实现

if(r1<r2) goto R

slt r1,r2,at

bne at,zero,R

伪指令-只可以使用push pop

寄存器表

2.子程序调用

寄存器使用规定（参数返回参数过多入栈）

栈高低位（高位开始，入栈则栈指针向低位走，存储的是内存中的地址）

push $r= addi $sp,$sp,-4; sw $r,0($sp);

过程调用时callee（ra,s)与caller(a,t(若有需要))分别需要保存的寄存器。

Q: int i该怎么翻译，float j呢？char k呢?

汇编时需要注意的几个：指针，数组，a[i]需i*4，出入栈，循环（初始化，for:判断条件 循环体 循环条件执行语句 j For Exit：），条件（比较不满足j ELSE; 满足条件的行动，ELSE:条件后的执行语句），递归（Fcout: 返回地址入栈，判断条件不满足则j R ;满足递归出口条件执行的语句，jr $ra,R：递归语句，参数入栈，新参数，jal Fcout

X86有什么考点吗【？】段逻辑地址与物理地址?

3.码

原码，移码，补码：已知原数求其码，已知码求其原数（注意是否大于8000) 2^N-1 可写为0x7FFG

（大于8000）移码：X-2^N-1,原码：2^n-1-X, 补码：2^N-X

小于8000：移码：X-2^N-1,原码：X, 补码：X

移码正负与原码、补码相反）

原码，反码，补码（2's complent)

4.加减乘除

加——

减——

乘——

除——

5.浮点运算

6.单时钟

指令的执行由时钟触发，如果考虑每个时钟，CPU执行一条指令，称为单时钟。

状态元件（先读后写两个时钟）——寄存器单元，一个时钟的延迟。读入：把这个值给自己，读出：把自己的值输出去

组合逻辑元件（输入直接决定输出）——逻辑门，译码器（一个输入多个输出但只有一个和其他所有输出端不同可访问），多路选择器（多个输入只有一个能输出），移位（连线），符号扩展（连线，有无符号重复符号位/0）

复杂模块：ALU及ALUop, 存储器（寄存器+译码器）

数据通道设计

信号处理（每个信号0和1对应的值，每条指令对应的信号值）

增加指令——设计操作码指令，增加必要组建，连线，设定控制信号，默认均为0。原来的1和0需要新增就加成10/11.

jal : ra=PC+4,j address ，把PC+4那边扯根线到ALU出口（write入口），Write寄存器新增入口31，新增信号MemtoReg(10),RegDst(10)

Bne: PC+=4，if（$r1!=$r2) goto LABEL,把zero和Branch拉出来做一个异或，新增信号Bne=1，branch=1，ALUsrc=0,ALUop=01【新增指令BNE咋画】

CPU时间计算——涉及Memory，ALU（取指令，运算，存储器）为2，涉及寄存器，回写为1。问题-效率低，存储器重复。

7.多时钟

每个部件在每个时钟，可以分别执行各自的任务，各主要部件之间设置寄存器存放临时中间结果。

不同的信号，每条指令在每个周期的信号，流程，所需周期数。（抄那张表）

新增指令ADDI【不会，等会试试】，BNE，JAlr

（addi $t1,\t2,immediate)

1. IF，PC+=4
2. ID，PC+ofs
3. 跟立即数相交
4. 写回

总线型CPU【待填】

三态门-高电平（1），低电平（0），高阻抗（隔断），

总线——一个时钟只能放一个信号，新来的被锁住，请求被屏蔽，输入无所谓，输出由三态门选择输出。

总线（控制总路线，数据总路线，地址总线）

存储器

DRAM（集中刷新，分散刷新，异步刷新）动态存储器会漏（主存），静态存储器（cache）

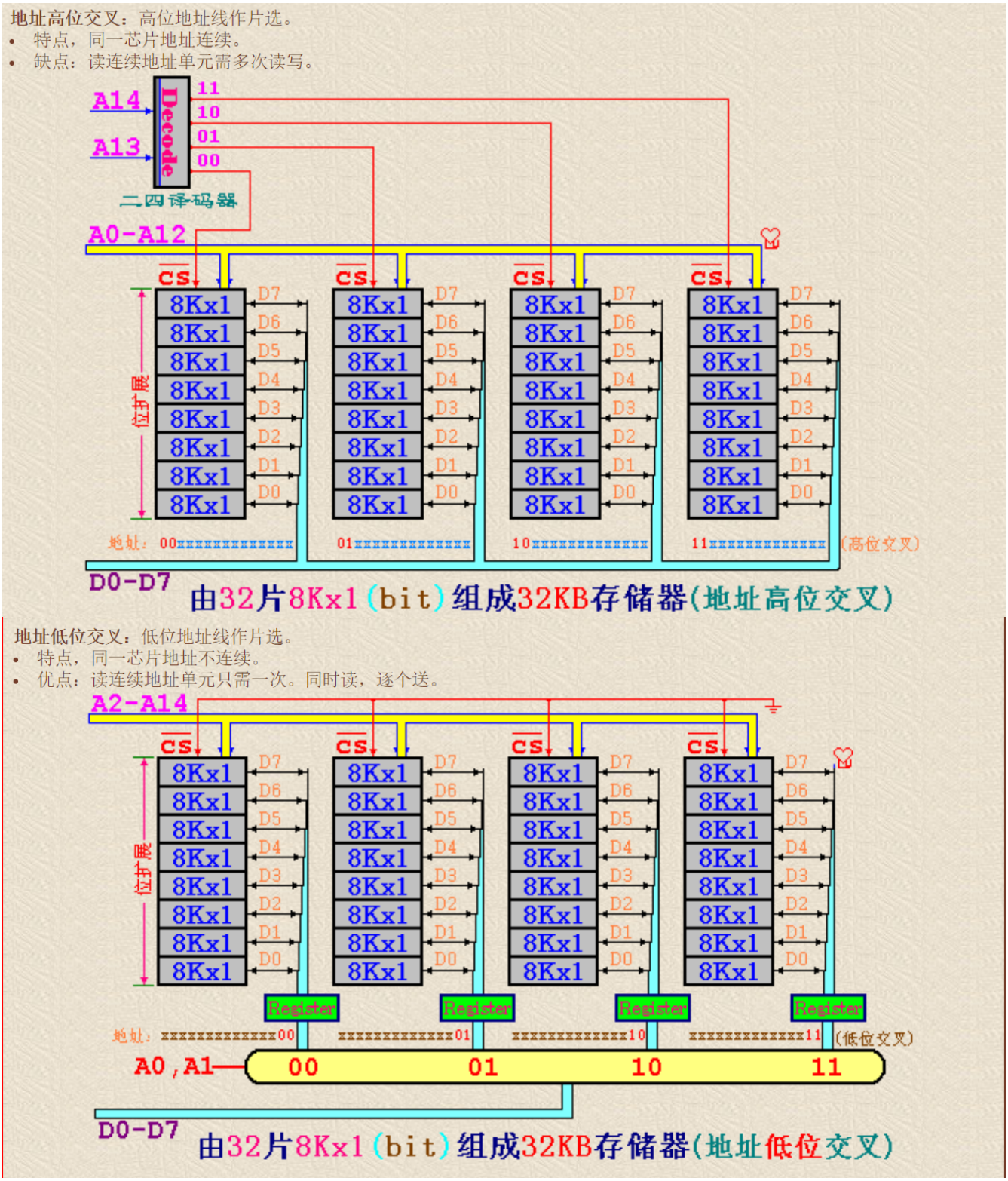
地址线（寻址范围），数据线（每个数多少个bit，数据位数）

Adr:20,dat:8,则为1MB, 32/32 则为4G*32bit

Cache： Index： valid tag（地址中除了确定index以外的值） data（即存储的数据，数据几位这里就是多少）

存储器地址-译码器，（二位地址译码，X,Y（存储阵列））

位扩展——（合并为存储字）将地址线以二维阵列方式排列，存储单元阵列以相同地址选中，每个二维阵列对应不同数据线。



对齐与不对齐

大头：高位字节位于地址小的字节(0x12345678 -> 78563412)

Cache原理

命中率，缺失率

差距在于：判断命中难易，Cache块分布是否均匀

空间局部性原理：按个位数将产生较少冲突

直接映射、组相连映射、全相连映射之**地址换算**（块号的映射）几组就看最后几位。

先计算物理块号=物理地址/块大小（最低几位），块内地址为物理地址mod块大小。Cache块号=物理块号 modCache组数（中间几位），TAG=物理块号 mod 组数（最高几位）。

直接映射：一组一块，4-way映射：一组4块，全映射：一组全部块。

TAG位数计算，需要空间大小计算=（块大小+TAG位数+1）*块数。其他不变，全>组>直接。

写策略：写回（Dirty），写通

替换策略：LRU，FIFO，LFU，Random

失配：强制，容量，相连（加大组相连）

虚拟内存

存储器和磁盘，速度数量及相差大，需要算法优化替换策略（LRU），并选择全相连。

地址转换——访问内存时的映射。物理页可以被两个虚拟地址共享—理解为指针。

虚拟地址=虚页号（比物理页数多，描述无容量限制的家乡）+页偏移（页大小），

页缺失——不在主存中需要去磁盘里，代价很高（数量级差距为10^5)->全相连减少失配率，并采用写回机制->项定位困难，设置页表（TLB）采取索引机制。

操作系统加载页表寄存器（页表首地址的寄存器）来指向希望激活的进程的页表而不是整个。

页表项32位，（有效位）+物理地址空间/页大小，其他用来存储别的保护信息（脏位，引用位）虚页号数量。

LRU替代策略——引用位（使用位）记录访问。

计算页表容量->页表项数=虚拟内存/页大小（虚页号数量），页表容量=页表项数*X字节/项。

页表包括物理页地址表和磁盘页地址表，逻辑上为一个，保存在两个独立数据结构中。

TLB——页表的cache,很小，全相连映射成本不会太高，缺失频繁，需要较低代价——随机选择替换表现。

一般框架：块放在哪（直接、组、全），如何找块（索引、索引+查找组内元素、查找所有cache项），缺失策略（随机、LRU），写操作处理（写通、写回）

强制缺失->块的第一次访问，增大块大小减少块数量，缺失代价增长，容量缺失->cache容纳不了所有块，增大cache容量，碰撞（除全相连）：多个块竞争一个组，提高相连度，延长访问时间。

总线（标准化）

系统功能部件间进行数据传送的公共通路。

内部——运算与寄存器，系统——CPU同存储器，I/O——中低速I/O设备间总线。

设备争用总线控制器，优先权/公平

流水线冒险

流水线冒险->下一个时钟周期中下一条指令不能执行。

结构：硬件不支持多条指令在统一时钟周期。MIPS可避免

数据：一条指令等待另一个指令完成而造成流水线暂停。

解决方法：前推（forwarding/bypassing)->从内部资源找你哥直接提前得到缺失的运算项。

控制、分支：决策依赖于一条指令的结果，而其他指令正在执行中

解决方法：

阻塞：分支指令阻塞要多一个时钟周期。

预测：预测分支未发生，发生时再阻塞。

I/O interface

程序查询方式：先检查外设状态，状态允许后进入数据I/O传送。

轮询：无定型简单输入输出，中断：随机性，混乱，如键盘鼠标，DMA：大量数据，声卡。

DMA：搬家队搬家，可停止任务，可穿插使用。

DMA间CPU工作方式：

1. 停止总线与存储器访问
2. 周期挪用（DMA读写外设的时间进行访问）
3. 轮流访问：DMA分成较小的周期，交替访问。

磁盘

Disk time=average seak time + average rotation delay +transfer time +controller overhead