性能因素 ①Throughput(Bandwidth):单位时间完成任务数量即吞吐量。CPU time=周期数×周期长度
②latency speed up = $\frac{T_{old}}{T_{new}}$

AMAT:平均内存访问时间

故障率 = $\frac{1}{MTTF}$

availability = $\frac{MTTF}{MTTF+MTTR}$

Pipelines
- single function:单功能,如只进行浮点加法
- Multi-function:多功能

Fit 10^9小时内出现的失效次数

- static pipeline:静态流水线同一时间只进行一种功能,如果切换功能要先清空当前任务
- Dynamic pipeline:同一时间处理不同指令,如加法后紧跟乘法

不同粒度:①Component level 部件级的流水线 ② Processor level ③ Inter-processor level:处理器间

指令依赖	冒险
- Data依赖	Data → RAW
- 名称依赖 - 反依赖	→ WAR
- 输出依赖	→ WAW
- 控制依赖	Branch
	structure

① Throughput (TP): $\frac{\text{指令数量}}{\text{执行时间}}$,把最长的流水线称 bottleneck segment(Δt), $TP_{max} = \frac{1}{\Delta t}$

(subdivision 瓶颈段分段执行, repetition 瓶颈段多个同功能部件并行)

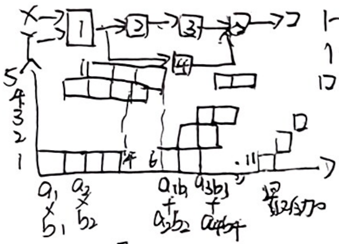
② speed up (SP): m: 流水线段数 n: 指令数 $Sp = \frac{n \times m}{m+n-1}$ ($n \gg m$时得m)

Forward: ALU+ALU:无nop,直接旁路回ID
load+store:添加一个Mux到Mem,不stall
load+ALU,stall一周期

③ efficiency (部件使用率,部件使用时间/总时间), $\eta = \frac{n}{m+n-1}$, $TP = \eta \cdot TP_{max}$

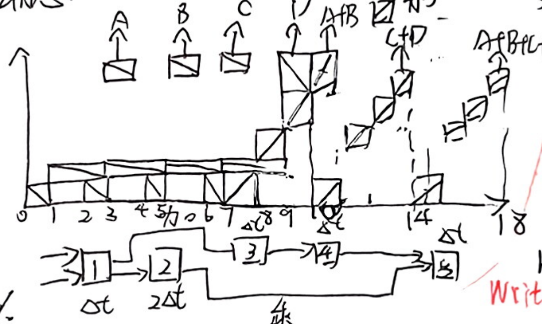
Latency:产生结果的指令和使用结果的指令之间的间隔周期
interval:两条连续指令间隔的周期

vector A(a_1, a_2, a_3, a_4)
vector B(b_1, b_2, b_3, b_4)
静态 算A·B(乘法完成才能开始加法)



$TP = \frac{7}{15\Delta t}$, $Sp = \frac{4\times3+3\times4}{15} = 1.6$, $\eta = 32\%$

动态: $TP = \frac{7}{18\Delta t}$, $Sp = \frac{4\times4+3\times4}{18}$ 乘 =1.56, $E = \frac{4\times4+3\times4}{5\times18}$ =0.31 (5个部件)



Memory

Block Address		Offset
tag	Index	

在Block中的位置

64个block,每个16B, byte address 1200 指向哪个block?
1200/16=75 75%64=11

Hit
Write Back+Write Allocate(Miss时先加载到Cache,写入Cache)
Miss
Write throught No-Write Allocate(Miss时直接写memory)

AMAT = Whole access Time / 访问次数 = Hit time + Miss Rate × Miss Penalty

①Compulsory Miss(冷启动,第一次访问某块时该块必然不命中)
②Capacity Miss:装不下
③Conflict Miss:比如Direct Mapped中不同块映射同一个位置

CPU Rate = 200MHz, Ideal CPI = 1.1
50% ALU, 30% ld/st, 20% control
miss penalty: 50 cycle
10% ld/st Miss
1% inst Miss
CPI? AMAT?

CPI = 1.1 + 10%×30%×50 + 1%×50 = 3.1
如果总共一条指令,则访问内存1.3次
AMAT = $\frac{1\times(1+0.01\times50)+0.3\times(1+0.1\times50)}{1.3}$ = 2.54

优化Cache:
① miss penalty↓:多级Cache, Read Miss优先, victim cache, critical word first
② miss Rate↓: block size, cache size↑, 组相联度↑, way预测
③ hit time↓ small simple cache
④ 并行↓miss Rate, penalty: 预取 non-blocking caches

虚拟内存:全相联, LRU, Write Back, TLB: Page Table的Cache, tag→VPN, data→PPN

ILP superscalar:2条及以上指令并行执行(多发射),静态:用编译器,动态:硬件 | super pipeline:超流水线,级数多
VLIW:把多条可以并行的指令打包成一条,然后在EX段并行执行

ScoreBoard
①Instruction Status Table
②Functional Unit Status Table
③Register Result Table
IS→RO→EX→WB
busy:是否使用
op:指令类型
Qj,Qk:源操作数来源的部件
Rj,Rk:是否Ready同时看
Qj,Qk是否为null来判断值是否准备好
Fj,Fk:源操作数寄存器
Fi:目的寄存器

	IS	RO	EX	WB
FLD F6,34(R2)	1	2	3	4
FLD F2,45(R3)	5	6	7	8
FMUL.D F0,F2,F4	6	9	10~19	20
FSUB.D F8,F2,F6	7	9	10~11	12
FDIV.D F10,F0,F6	8	21	22~61	62
FADD.D F6,F8,F2	13	14	15~16	22

	IS	EX	WB
FLD F6,34(R2)	1	3	4
FLD F2,45(R3)	2	4	5
FMUL.D F0,F2,F4	3	6~15	16
FSUB.D F8,F2,F6	4	6~7	8
FDIV.D F10,F0,F6	5	17~56	57
FADD.D F6,F8,F2	6	9~10	11

Tomasulo
Reservation Stations
Qj,Qk:保留站编号(空表示就绪)
op
Vj,Vk:源操作数的值
A:Address for ld/st

Tomasulo with ROB 保留站有Dest
IS→EX→WB→Commit
指令发射时需要保留站和ROB都有空位,一个ROB记录有OP,目标寄存器,值,就绪位
指令发射时把信息送到ROB,WB时值送到ROB中,经过Bus
Qj,Qk变为ROB序号,Register也记录ROB序号

	IS	EX	WB	Commit	Ready time
FLD F6,34(R2)	1	3	4	5	
FLD F2,45(R3)	2	4	5	6	
FMUL.D F0,F2,F4	3	6~15	16	17	
FSUB.D F8,F6,F2	4	6~7	8	18→前一条Commit后才commit	
FDIV.D F10,F0,F6	5	17~56	57	58	
FADD.D F6,F8,F2	6	9~10	11	59	

ROB	Busy	Inst	Status	Dest	Value
1	no	FLD F6,34(R2)	commit	F6	Mem[load1]
2	no	FLD	commit	F2	Mem[load2]
3	Y	Mul	EX	F0	
4	Y	SUB	EX	F8	
5	Y	DIV	IS	F10	
6	Y	ADD	IS	F6	

DLP

Flynn: SISD - 冯·诺依曼
SIMD (DLP) < vector processor / Array processor
MIMD (TLP)
MISD (不存在)

$D = A \times (B+C)$
① Horizontal process: 算 $d_1 = a_1 \times (b_1 + c_1)$, $d_2 = a_2 \times (b_2 + c_2)$ ...
② vertical process: 先算 $K = B+C$，再算 $D = A \times K$
③ H and V process: 分组，组内纵向，组间横向

Chaining: 解决RAW相关指令放一起

convey: 一组可以在同一个时钟中执行的向量指令，无结构冒险
chime: 执行一个convey的时间，执行m个convey需m次chime，向量长度为n时需m×n个cycle

DAXPY (Double Precision a×X+Y)

RISC-V (scalar):
```
fld f0,a
addi x28,x5,256
Loop: fld f1,0(x5)
fmul.d f1,f1,f0   //a*X[i]
fld f2,0(x6)
fadd.d f2,f2,f1
fsd f2,0(x6)      //a*X[i]+Y[i]
addi x5,x5,8
addi x6,x6,8
bne x28,x5,Loop
```

RV64V:
```
vsetdcfg 4*FP64
fld f0,a
vld v0,x5   //vector X
vmul v1,v0,f0
vld v2,x6
vadd v3,v1,v2
vst v3,x6
vdisable
```

Cube: 对第i位取反
$PM2_{+i}(j) = (j + 2^i) \bmod N$, $PM2_{-i}(j) = (j - 2^i) \bmod N$
Shuffle($P_{n-1} \cdots P_1 P_0$) = $P_{n-2} \cdots P_1 P_0 P_{n-1}$

vector processor 包含 ① Vector Reg ② Vector FU ③ vector Ld/St ④ Scalar Reg

① 无跨迭代依赖，直接并行
for(i=999;i>=0;i=i-1)
X[i]=X[i]+s;

② 有跨迭代依赖，可消除：
for(i=0;i<100;i++){
A[i]=A[i]+B[i];
B[i+1]=C[i]+D[i];}
不比B[i] B[i+1]先出现，则依赖
A[0]=A[0]+B[0];
→ for(i=0;i<99;i++){
B[i+1]=C[i]+D[i];
A[i+1]=A[i+1]+B[i+1];}
B[100]=C[99]+D[99];

③ 无法消除：for(i=0;i<100;i++){
A[i+1]=A[i]+C[i];
B[i+1]=B[i]+A[i+1];}

<<<2,128>>> 2个block 每个128 thread
GPU是多线程SIMD，grid里有block，block里有thread；wrap是一组thread
数组偏移 共享内存 私有内存

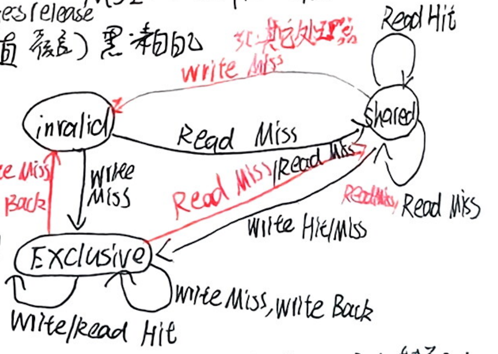
TLP

SMP/CSM/UMA：集中式共享内存：所有处理器共享同一物理内存
NUMA/DSM：分布式共享内存，每个处理器有本地内存

TSO：消除W(a)→R(b)
PSO：消除W(a)→W(b)
WO：消除R(a)→R(b)，R(a)→W(b)
release一致性：只保证关键的操作有序用acquire/release

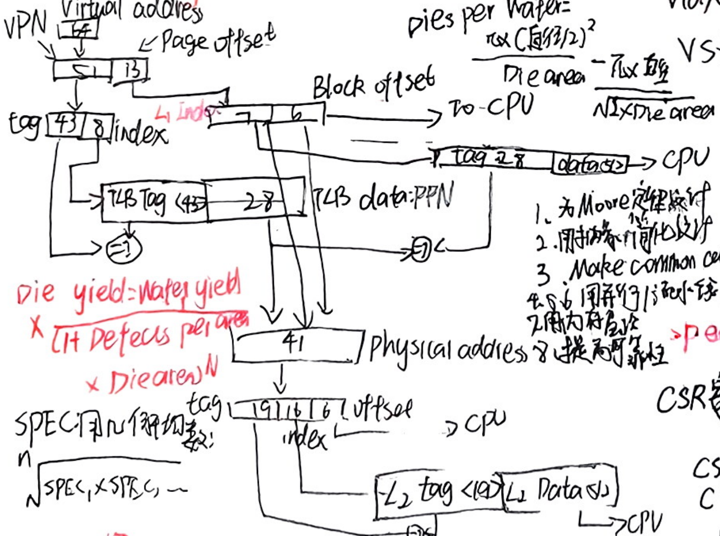
Cache Coherence：读取可以返回什么值
① P write(x)，马上read(x)，且此期间无其它处理器写X，读返回写的值
② A read(x)在B write(x)后，且经过足够时间，无其它处理器写X，则返回write(B)的值
③ 对同一位置X的所有写入操作在任一处理器看来都以相同时间顺序发生
Consistency：写的值什么时候可以被读返回

MSI：Modified Shared Invalid 写后读要(Cache 更新)



vldx v1,x5,v0 //load A[K[i]]
vstx v1,x5,v0 //store A[K[i]]

dies per wafer = $\frac{\pi \times (直径/2)^2}{Die\ area} - \frac{\pi \times 直径}{\sqrt{2 \times Die\ area}}$



Die yield = Wafer yield × $(1 + Defects\ per\ area \times Die\ area)^{-N}$

1. 为Moore定律设计
2. 用抽象简化设计
3. Make common case fast
4. 5. 6. 用并行/流水线/预测提高性能
7. 层次存储
8. 提高可靠性

A[a×i+b]=...
...=A[c×i+d]
存在依赖，当：$a \times j + b = c \times k + d$
若存在依赖，$gcd(a,c)$ 整除 $d-b$

Speed up = $\frac{1}{1 - Frac_{enhanced} + \frac{Frac_{enhanced}}{Speed\ up}}$

CSR寄存器：mstatus, mie, mip, mcause, mepc, mtvec, mtval
csrrw (read write), csrrs (read, set)
csrrc (read, clear)
mtval：异常地址或异常指令内容
mtvec：异常处理程序入口

MESI：Exclusive：Data仅在当前Cache中独有且未被修改，写不用发出Invalidate
MOESI：Owned，该数据被当前Cache拥有且与内存中的值不同，其他缓存中，其它缓存从该缓存处获取数据
MOESIF：总线Read Miss, Modified→Forward
目录中目录项状态：Modified Shared Uncached
数据块状态：MSI

SPEC用n个比值的几何平均数：$\sqrt[n]{SPEC_1 \times SPEC_2 \times \cdots}$

ROB，双发射 双提交

	IS	EX	Mem	WB	Com
LD R2,0(R1)	1	2	3	4	5
ADDI R2,R2,1	1	5		6	7
SD R2,0(R1)	2	3			7
ADDI R1,R1,4	2	3		4	8
BNE R2,R3,Loop	3	7			8
LD R2,0(R1)	4	5	6	7	9
ADDI R2,R2,1	4	8		9	10
SD R2,0(R1)	5	6			10
ADDI R1,R1,4	5	6		7	11
BNE R2,R3,Loop	6	10			11

index+offset 几位决定隔多少字节映射到一起

原子性同步指令
① Atomic exchange：寄存器的值与内存中的一个值交换
② Test and set：测试是否为新值，若是则设为新值
③ Fetch and increment：不断++，return i++

$D = A \times (B+C)$，add 6 cycles，ld 6 cycles，mul 7 cycles
$V_3 \leftarrow A$，$V_2 \leftarrow V_0+V_1$，$V_4 \leftarrow V_2 \times V_3$
① 串行：(1+6+1)+N−1+(1+6+1)+N−1+(1+7+1)+N−1 = 3N+22
② 前两个并行：max{(1+6+1)+N−1, (1+6+1)+N−1}+1+7+1+N−1 = 2N+15
③ 前两个并行+chain：max(1+6+1, 1+6+1)+1+7+1+N−1 = N+16

Sequential Consistency
① 处理器内部有序
② 不同处理器操作交错

Relaxed Consistency
① 允许读和写无序，使用同步操作
X→Y，Y必须在X之后完成
relax W→R，允许读在写之前完成

GPU 10个SIMD处理器，8 lanes，80%线程活跃，70%无停顿，20% ld/st，issue rate 0.85，1.5GHz，问GFLOP/s？
$1.5GHz \times 0.8 \times 0.85 \times 0.7 \times 10 \times 8 = 57.12$