Interesting Numbers {single,double}

Description	*e xp*	*frac*	*Numeric Value*
Zero	**00…00**	**00…00**	**0.0**
Smallest Pos. Denorm.	**00…00**	**00…01**	$2^{-\{23,52\}} \times 2^{-\{126,1022\}}$
▪ Single ≈ 1.4×10^{-45}			
▪ Double ≈ 4.9×10^{-324}			
Largest Denormalized	**00…00**	**11…11**	$(1.0 - \varepsilon) \times 2^{-\{126,1022\}}$
▪ Single ≈ 1.18×10^{-38}			
▪ Double ≈ 2.2×10^{-308}			
Smallest Pos. Normalized	**00…01**	**00…00**	$1.0 \times 2^{-\{126,1022\}}$
▪ Just larger than largest denormalized			
One	**01…11**	**00…00**	**1.0**
Largest Normalized	**11…10**	**11…11**	$(2.0 - \varepsilon) \times 2^{\{127,1023\}}$
▪ Single ≈ 3.4×10^{38}			
▪ Double ≈ 1.8×10^{308}			

bwlq
1248
8 16 32 64
char short int long long
movzbl

Type	Form	Operand value	Name
Immediate	$\$Imm$	Imm	Immediate
Register	$\mathtt{r}_a$	$\mathsf{R}[\mathtt{r}_a]$	Register
Memory	Imm	$\mathsf{M}[Imm]$	Absolute
Memory	$(\mathtt{r}_a)$	$\mathsf{M}[\mathsf{R}[\mathtt{r}_a]]$	Indirect
Memory	$Imm(\mathtt{r}_b)$	$\mathsf{M}[Imm + \mathsf{R}[\mathtt{r}_b]]$	Base + displacement
Memory	$(\mathtt{r}_b,\mathtt{r}_i)$	$\mathsf{M}[\mathsf{R}[\mathtt{r}_b] + \mathsf{R}[\mathtt{r}_i]]$	Indexed
Memory	$Imm(\mathtt{r}_b,\mathtt{r}_i)$	$\mathsf{M}[Imm + \mathsf{R}[\mathtt{r}_b] + \mathsf{R}[\mathtt{r}_i]]$	Indexed
Memory	$(,\mathtt{r}_i,s)$	$\mathsf{M}[\mathsf{R}[\mathtt{r}_i] \cdot s]$	Scaled indexed
Memory	$Imm(,\mathtt{r}_i,s)$	$\mathsf{M}[Imm + \mathsf{R}[\mathtt{r}_i] \cdot s]$	Scaled indexed
Memory	$(\mathtt{r}_b,\mathtt{r}_i,s)$	$\mathsf{M}[\mathsf{R}[\mathtt{r}_b] + \mathsf{R}[\mathtt{r}_i] \cdot s]$	Scaled indexed
Memory	$Imm(\mathtt{r}_b,\mathtt{r}_i,s)$	$\mathsf{M}[Imm + \mathsf{R}[\mathtt{r}_b] + \mathsf{R}[\mathtt{r}_i] \cdot s]$	Scaled indexed


63
31
15
7
0
%rax
%eax
%ax
%al
Return value
%rbx
%ebx
%bx
%bl
Callee saved
%rcx
%ecx
%cx
%cl
4th argument
%rdx
%edx
%dx
%dl
3rd argument
%rsi
%esi
%si
%sil
2nd argument
%rdi
%edi
%di
%dil
1st argument
%rbp
%ebp
%bp
%bpl
Callee saved
%rsp
%esp
%sp
%spl
Stack pointer
%r8
%r8d
%r8w
%r8b
5th argument
%r9
%r9d
%r9w
%r9b
6th argument
%r10
%r10d
%r10w
%r10b
Caller saved
%r11
%r11d
%r11w
%r11b
Caller saved
%r12
%r12d
%r12w
%r12b
Callee saved
%r13
%r13d
%r13w
%r13b
Callee saved
%r14
%r14d
%r14w
%r14b
Callee saved
%r15
%r15d
%r15w
%r15b
Callee saved

SetX	Condition	Description
sete	ZF	Equal / Zero
setne	~ZF	Not Equal / Not Zero
sets	SF	Negative
setns	~SF	Nonnegative
setg	~(SF^OF) &~ZF	Greater (Signed)
setge	~(SF^OF)	Greater or Equal (Signed)
setl	(SF^OF)	Less (Signed)
setle	(SF^OF) | ZF	Less or Equal (Signed)
seta	~CF&~ZF	Above (unsigned)
setb	CF	Below (unsigned)

Format	*Computation*	
addq	*Src,Dest*	Dest = Dest + Src
subq	*Src,Dest*	Dest = Dest – Src
imulq	*Src,Dest*	Dest = Dest * Src
salq	*Src,Dest*	Dest = Dest << Src
sarq	*Src,Dest*	Dest = Dest >> Src
shrq	*Src,Dest*	Dest = Dest >> Src
xorq	*Src,Dest*	Dest = Dest ^ Src
andq	*Src,Dest*	Dest = Dest & Src
orq	*Src,Dest*	Dest = Dest | Src

incq *Dest*	*Dest = Dest + 1*
decq *Dest*	*Dest = Dest – 1*
negq *Dest*	*Dest = – Dest*
notq *Dest*	*Dest = ~Dest*

Address Computation Instruction

- `leaq` *Src*, *Dst*
  - *Src* is address mode expression
  - Set *Dst* to address denoted by expression

- **Uses**
  - Computing addresses without a memory reference
    - E.g., translation of `p = &x[i];`
  - Computing arithmetic expressions of the form x + k*y
    - k = 1, 2, 4, or 8

- **Example**

```
long m12(long x)
{
  return x*12;
}
```

Converted to ASM by compiler:

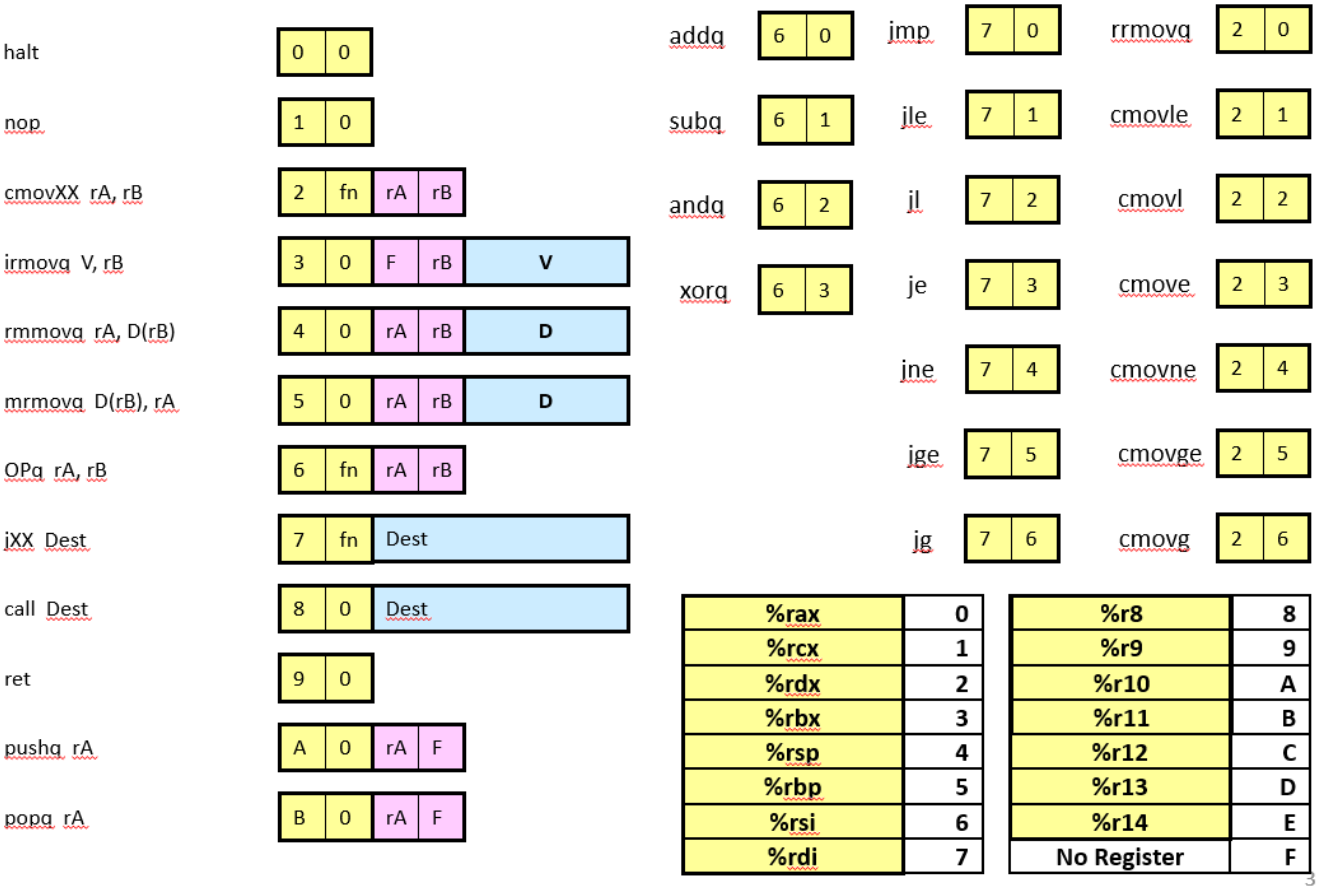
```
leaq (%rdi,%rdi,2), %rax # t <- x+x*2
salq $2, %rax            # return t<<2
```


`cmp a,b b-a`

`test a,b a&b`

jX	Condition	Description
jmp	1	Unconditional
je	ZF	Equal / Zero
jne	~ZF	Not Equal / Not Zero
js	SF	Negative
jns	~SF	Nonnegative
jg	~(SF^OF)&~ZF	Greater (Signed)
jge	~(SF^OF)	Greater or Equal (Signed)
jl	(SF^OF)	Less (Signed)
jle	(SF^OF)|ZF	Less or Equal (Signed)
ja	~CF&~ZF	Above (unsigned)
jb	CF	Below (unsigned)

Y86-64 instruction set



弄反

分成 n 块，每块是 max

ps = $10^{-12}ps$

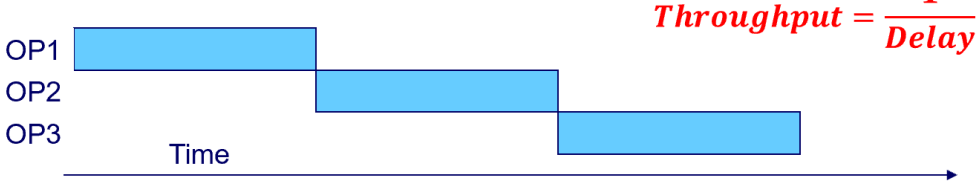
IPS

MIPS /1e6

Pipeline Diagrams

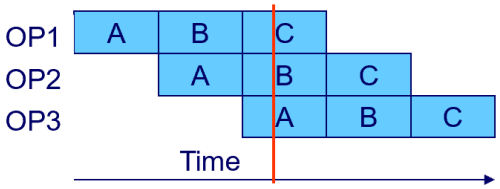
Unpipelined

$$Throughput = \frac{1}{Delay}$$



- Cannot start new instruction until previous one completes

3-Way Pipelined



$$Throughput = \frac{1}{Delay} \times N$$

- Up to 3 instructions running simultaneously

Latency 一个的长度
delay n 个的长度
throughput = n/delay = 1 / latency

Nonvolatile Memories

- DRAM and SRAM are **volatile memories**
  - Lose information if powered off
- **Nonvolatile memories** retain value even if powered off
  - Read-only memory (ROM): programmed during production
  - Programmable ROM (PROM): can be programmed once
  - Erasable PROM (EPROM): can be erased 1000 times
  - Electrically eraseable PROM (EEPROM): can be erased 100,000 times
  - Flash memory: EEPROMs. with partial (block-level) erase capability
- Uses for Nonvolatile Memories
  - Firmware programs stored in a ROM (BIOS, controllers for disks, network cards, graphics accelerators, security subsystems,…)
  - Solid state disks (thumb drives, smart phones, mp3 players, tablets, laptops,…)
  - Disk caches

A Disk Drive

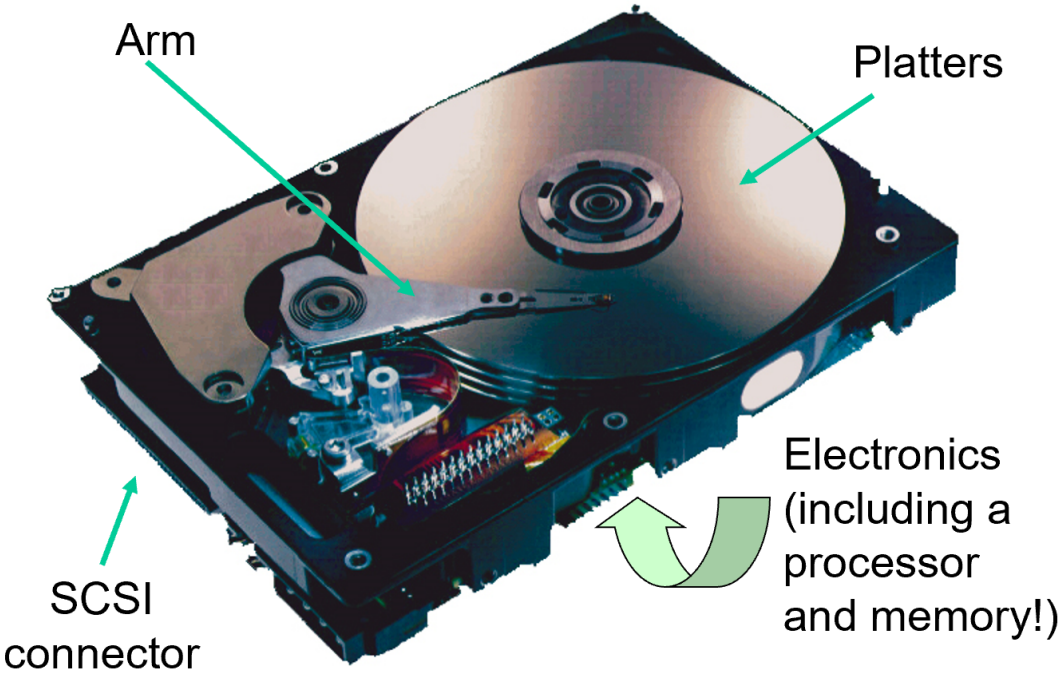
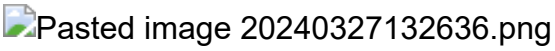
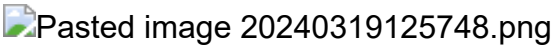
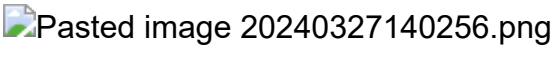
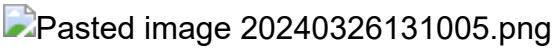


Image courtesy of Seagate Technology

硬盘有很多 Platter (拼盘)，每个盘有两个面 (Surface)，每个面分成若干 tracks (圆环)，分成若干段 (sectors)
Disk controller 控制
DMA Direct Memory Acess, Disk → Main Memory
Solid State Disks: 闪存 Block Page
 Pasted image 20240327132636.png

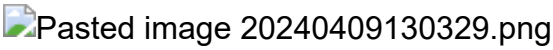
Hit: 已经在 cache
miss: 不在
算地址记得*一个元素的byte数（4）
FIFO
LRU：每次在 cache 里拖到最后
 Pasted image 20240319125748.png
同步异常 Traps 故意可恢复, Faults 不故意可以恢复, Aborts 不故意不可恢复
 Pasted image 20240327140256.png

```
fork()
exit()
wait() 收割reaping
waitpid()
getpid()
getppid()
```

 Pasted image 20240326131005.png
N (virtual, disk) > M (physical, dram)
共享偏移量
page table
page hit: 在 m (physical memory / dram 里)
换一个驱逐然后 page hit
如果工作集>主内存 爆了
PTE (page table entries)
MMU (Memory management unit)
Page table base register (PTBR)

VPN PPN (Viertual Physical page number / offset)
TLBI TLBT.

Lec11


父子共享一个（描述符）但是同一个程序多个是并行