

CSE 351 Reference Sheet (Final)

Binary	Decimal	Hex
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
1	2	4	8	16	32	64	128	256	512	1024

SI Size	Prefix	Symbol	IEC Size	Prefix	Symbol
10^3	Kilo-	K	2^{10}	Kibi-	Ki
10^6	Mega-	M	2^{20}	Mebi-	Mi
10^9	Giga-	G	2^{30}	Gibi-	Gi
10^{12}	Tera-	T	2^{40}	Tebi-	Ti
10^{15}	Peta-	P	2^{50}	Pebi-	Pi
10^{18}	Exa-	E	2^{60}	Exbi-	Ei
10^{21}	Zetta-	Z	2^{70}	Zebi-	Zi
10^{24}	Yotta-	Y	2^{80}	Yobi-	Yi

Sizes

C type	Suffix	Size
char	b	1
short	w	2
int	l	4
long	q	8

IEEE 754 FLOATING-POINT STANDARD

Value: $\pm 1 \times \text{Mantissa} \times 2^{\text{Exponent}}$

Bit fields: $(-1)^S \times 1.M \times 2^{(E-\text{bias})}$

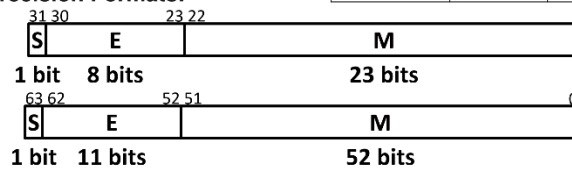
where Single Precision Bias = 127,

Double Precision Bias = 1023.

IEEE 754 Symbols

E	M	Meaning
all zeros	all zeros	± 0
all zeros	non-zero	\pm denorm num
1 to MAX-1	anything	\pm norm num
all ones	all zeros	$\pm \infty$
all ones	non-zero	NaN

IEEE Single Precision and Double Precision Formats:



Assembly Instructions

mov a, b	Copy from a to b.
movs a, b	Copy from a to b with sign extension. Needs two width specifiers.
movz a, b	Copy from a to b with zero extension. Needs two width specifiers.
lea a, b	Compute address and store in b. <i>Note: the scaling parameter of memory operands can only be 1, 2, 4, or 8.</i>
push src	Push <i>src</i> onto the stack and decrement stack pointer.
pop dst	Pop from the stack into <i>dst</i> and increment stack pointer.
call <func>	Push return address onto stack and jump to a procedure.
ret	Pop return address and jump there.
add a, b	Add from a to b and store in b (and sets flags).
sub a, b	Subtract a from b (compute $b-a$) and store in b (and sets flags).
imul a, b	Multiply a and b and store in b (and sets flags).
and a, b	Bitwise AND of a and b, store in b (and sets flags).
sar a, b	Shift value of b <i>right (arithmetic)</i> by a bits, store in b (and sets flags).
shr a, b	Shift value of b <i>right (logical)</i> by a bits, store in b (and sets flags).
shl a, b	Shift value of b <i>left</i> by a bits, store in b (and sets flags).
cmp a, b	Compare b with a (compute $b-a$ and set condition codes based on result).
test a, b	Bitwise AND of a and b and set condition codes based on result.
jmp <label>	Unconditional jump to address.
j* <label>	Conditional jump based on condition codes (<i>more on next page</i>).
set* a	Set byte based on condition codes.

Conditionals

Instruction	(op) s, d	test a, b	cmp a, b
je sete “Equal”	d (op) s == 0	b & a == 0	b == a
jne setne “Not equal”	d (op) s != 0	b & a != 0	b != a
js sets “Sign” (negative)	d (op) s < 0	b & a < 0	b-a < 0
jns setns (non-negative)	d (op) s >= 0	b & a >= 0	b-a >= 0
jg setg “Greater”	d (op) s > 0	b & a > 0	b > a
jge setge “Greater or equal”	d (op) s >= 0	b & a >= 0	b >= a
jl setl “Less”	d (op) s < 0	b & a < 0	b < a
jle setle “Less or equal”	d (op) s <= 0	b & a <= 0	b <= a
ja seta “Above” (unsigned >)	d (op) s > 0U	b & a > 0U	b-a > 0U
jb setb “Below” (unsigned <)	d (op) s < 0U	b & a < 0U	b-a < 0U

Registers

Name	Convention	Name of “virtual” register		
		Lowest 4 bytes	Lowest 2 bytes	Lowest byte
%rax	Return value – Caller saved	%eax	%ax	%al
%rbx	Callee saved	%ebx	%bx	%bl
%rcx	Argument #4 – Caller saved	%ecx	%cx	%cl
%rdx	Argument #3 – Caller saved	%edx	%dx	%dl
%rsi	Argument #2 – Caller saved	%esi	%si	%sil
%rdi	Argument #1 – Caller saved	%edi	%di	%dil
%rsp	Stack Pointer	%esp	%sp	%spl
%rbp	Callee saved	%ebp	%bp	%bpl
%r8	Argument #5 – Caller saved	%r8d	%r8w	%r8b
%r9	Argument #6 – Caller saved	%r9d	%r9w	%r9b
%r10	Caller saved	%r10d	%r10w	%r10b
%r11	Caller saved	%r11d	%r11w	%r11b
%r12	Callee saved	%r12d	%r12w	%r12b
%r13	Callee saved	%r13d	%r13w	%r13b
%r14	Callee saved	%r14d	%r14w	%r14b
%r15	Callee saved	%r15d	%r15w	%r15b

C Functions

```
void* malloc(size_t size):
Allocate size bytes from the heap.

void* calloc(size_t n, size_t size):
Allocate n*size bytes and initialize to 0.

void free(void* ptr):
Free the memory space pointed to by ptr.

size_t sizeof(type):
Returns the size of a given type (in bytes).

char* gets(char* s):
Reads a line from stdin into the buffer.

pid_t fork():
Create a new child process (duplicates parent).

pid_t wait(int* status):
Blocks calling process until any child process
exits.

int execv(char* path, char* argv[]):
Replace current process image with new image.
```

Virtual Memory Acronyms

MMU	Memory Management Unit	VPO	Virtual Page Offset	TLBT	TLB Tag
VA	Virtual Address	PPO	Physical Page Offset	TLBI	TLB Index
PA	Physical Address	PT	Page Table	CT	Cache Tag
VPN	Virtual Page Number	PTE	Page Table Entry	CI	Cache Index
PPN	Physical Page Number	PTBR	Page Table Base Register	CO	Cache Offset