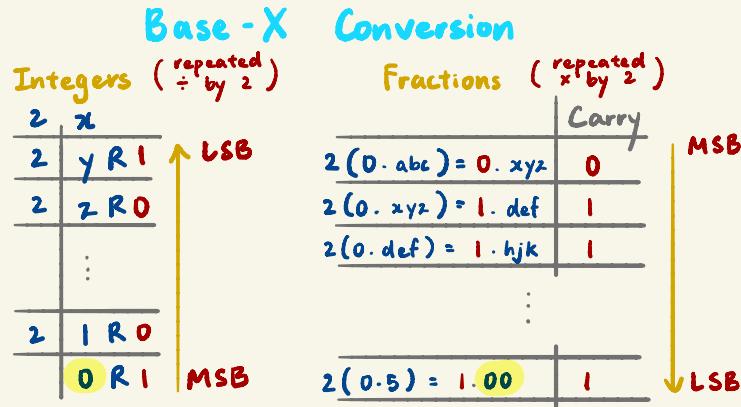


Data Representation & Number Systems



Binary \rightarrow Octal / Hexadecimal

$$(x)_8 : 010 \mid 0000 \ 0110 \ 0000 = 2014_8$$

$$(x)_{16} : 010 \mid 0000 \ 0110 \ 0000 = 2060_{16}$$

Data

bit (b): 0, 1
byte (B): 8b
word: 32b / 4B

N bits: 2^N values

M values: $\lceil \log_2 M \rceil$ bits

Integers

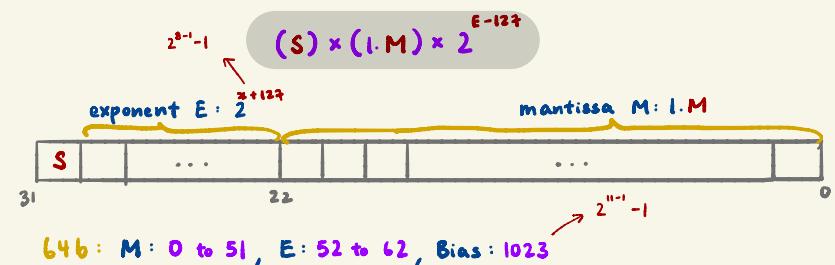
Rep	$+ \rightarrow -$	Range	Zeros
S-M.	Flip MSB	$-(2^{n-1}), 2^{n-1}-1$	$+/-$
ls	Flip ALL	$-(2^{n-1}), 2^{n-1}-1$	$+/-$
2s	\leftarrow till first 1, flip rest	$-2^{n-1}, 2^{n-1}-1$	+

Overflow $A + B : (MSB_A == MSB_B) \neq MSB_{A+B}$

ls: Add carry to LSB_{A+B} , check for overflow

2s: Ignore carry, check for overflow

IEEE - 754



C Programming

Primitives

Type	Size (b)	Range	
		Unsigned	Signed
char	8	[0, 255]	[-128, 127]
short	8	[0, 255]	[-128, 127]
int	32	[0, $2^{31}-1$]	[- 2^{31} , $2^{31}-1$]
long	64	[0, $2^{63}-1$]	[- 2^{63} , $2^{63}-1$]
float	32	-	-
double	64	-	-

Casting

(cast_type) var



Auto-promotion/demotion

```
2 / 5 == 0;
2.0 / 5 == 0.4;
int x = 0.4; // → 0
double y = 5; // → 5.0
```

Pointers

```
int x=5;
int *y = &x;
print(x); // 5
print(*y); // 5
print(&x); // addr(x)
print(y); // addr(x)
```

Pointer declaration
Dereference
Go to address of

Parameter-Passing

By-value : primitives (int, char,...), structures

By-address: arrays (strings) \rightarrow [], " "

scanf

```
int x;
scanf("%d", &x); // address
```

Arrays

```
int arr[5] = {1, 2, 3, 4, 5}
```

i	Value	Address
0	1	0x100
1	2	0x104
2	3	0x108
3	4	0x112
4	5	0x116
⋮	⋮	⋮

```
#(arr+2) == arr[2];
arr+2 == &(arr[2]);
arr == &arr[0];
```

Can be passed as parameter,
but not as return type

Strings

"Hello": char str[6] = "Hello"; string terminator
// {'H', 'e', 'l', 'l', 'o', '\0'}

Structures

```
struct struct_name {
    datatype var1; // one or more fields
    [datatype var2;]
}
```

Defining a new structure

Ways to instantiate

```
struct_name str1 = {1}; ← ==
```

Ways to reference field

```
struct_name *p = &str1; // pointer of struct-name
(*p).var1 = 1; ← ==
```

```
p → var1 = 1; ← ==
```

MIPS

Key Pitfalls

Load 32b immediate

```
Lui $sp, imm[32:16]
ori $sp, imm[15:0]
```

Branch-equal immediate

Imm no. of instructions:

AFTER next instruction

Imm size: 16b $\rightarrow 2^{16}-1$

$\therefore 2^{16}$ instructions from beq/bne

J-format immediate

Imm: 26b 'address'

Word-aligned:

- 2 LSB disregarded

- 4 MSB disregarded

\therefore Imm: Addr [27:2]

Branching

HLL code	if ($x < y$) { next; } branch;	if ($x \leq y$) { next; } branch;	if ($x > y$) { next; } branch;	if ($x \geq y$) { next; } branch;	if ($x == y$) { next; } branch;	if ($x != y$) { next; } branch;
Possible x and y	$x < y$	$x = y$	$x > y$	$x < y$	$x = y$	$x > y$
Truth values for above	TRUE	FALSE	FALSE	TRUE	TRUE	FALSE
slt j x y	1 0 0	1 0 0	1 0 0	1 0 0	1 0 0	1 0 0
slt k y x	0 0 1	0 0 1	0 0 1	0 0 1	0 0 1	0 0 1
beq j 0 branch	Next	Branch	Branch	Next	Branch	Branch
bne j 0 branch	Branch	Next	Next	Branch	Next	Next
beq k 0 branch	Branch	Branch	Next	Branch	Branch	Next
bne k 0 branch	Next	Next	Branch	Next	Next	Branch
beq x y branch	Next	Next	Branch	Next	Next	Branch
bne x y branch	Branch	Next	Branch	Branch	Next	Branch

Encoding

R	6b	5b	5b	5b	5b	6b
	opcode	rs	rt	rd	shamt	funct
	31 26 25	26 25 21 20	21 20 16 15	16 15 11 10	11 10 6 5	0
I	6b	5b	5b	5b	5b	6b
	opcode	rs	rt	Imm		
	31 26 25	26 25 21 20	21 20 16 15	0		
J	6b	5b	5b	5b	5b	6b
	opcode	Target addr				
	31 26	0				

Processors

Control Signals

	EX				MEM			WB	
	RegDst	ALUSrc	ALUOp		Mem Read	Mem Write	Branch	MemTo Reg	Reg Write
			op1	op0					
R-type	1	0	1	0	0	0	0	0	1
lw	0	1	0	0	1	0	0	1	1
sw	X	1	0	0	0	1	0	X	0
beq	X	0	0	1	0	0	1	X	0

ALU Control

ALUcontrol	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	sllt
1100	NOR

