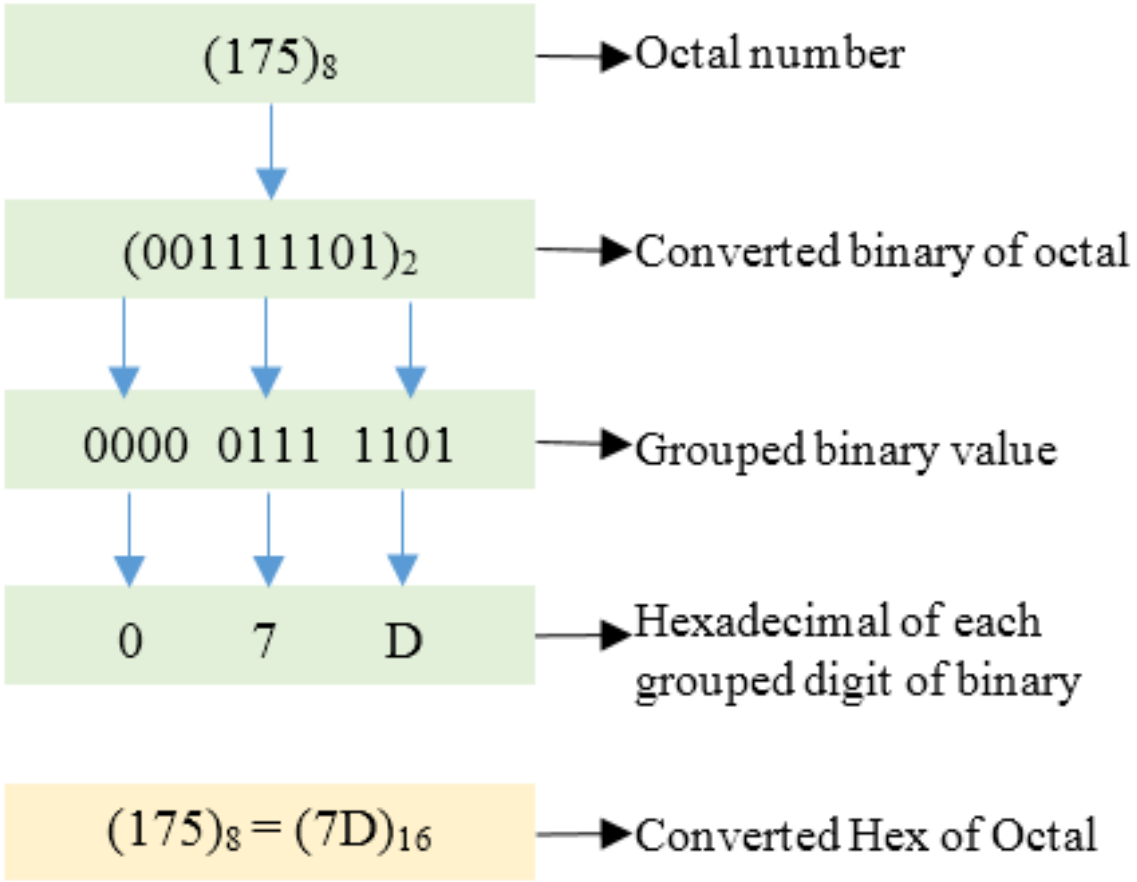
Tutorial3.1

Bitwise logical instructions

Lei Wang

lei.wang2@ucalgary.ca

binary, Octal, Hexadecimal



Binary	Octal	Decimal	Hexadecimal
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	10	A
1011	13	11	B
1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F

Representations in C

```
#inclue <stdio.h>
int main(int argc, char *argv[])
{
    int v3 = 0xFEEDDAD; //hexadecimal literal
    int v4 = 0b10010011; //binary literal
    int v5 = 022222222;  //octal literal

    printf("v3 = %d, 0x%x, 0%o\n", v3, v3, v3);
    printf("v4 = %d, 0x%x, 0%o\n", v4, v4, v4);
    printf("v5 = %d, 0x%x, 0%o\n", v5, v5, v5);
}
```

Bitwise Logical Instructions

- And
  - and Xd, Xn, Xm
  - ands Xn, Xm sets or clears N and Z flags according to the results(V and C are always cleared)
- OR
  - orr Xd, Xn, Xm
- XOR
  - eor Xd, Xn, Xm
- Bit Clear(AND NOT)
  - bic Xd, Xn, Xm

Bitwise Logical Instructions

- OR NOT
  - orn Xd, Xn, Xm
- NOT
  - mvn Xd, Xm
- EOR NOT
  - eon Xd, Xn, Xm

Truth table

x	y	AND	OR	XOR	BitClear	OR NOT	EOR NOT
0	0	0	0	0	0	1	1
0	1	0	1	1	0	0	0
1	0	0	1	1	1	1	0
1	1	1	1	0	0	1	1

Coding practice —bitwise NOT

- C

```c
#include <stdio.h>

void main(int argc, char *argv[]) {
    char a = 0b1001;

    printf("a=0x%x, ~a=0x%x\n", a, ~a);
}
```

- Assembly

```
fmt: .string "a=0x%x, ~a=0x%x\n"
    .balign 4
    .global main

main:
    stp  x29, x30, [sp, -16]!
    mov  x29, sp
    ldr  x0, =fmt
    mov  x1, 0b1001
    mvn  x2, x1                 // Move and NOT
    bl printf

    ldp x29, x30, [sp], 16
    ret
```

Coding practice —Bitwise OR

- C

```c
#include <stdio.h>

void main(int argc, char *argv[]) {
    char a = 0b1001;
    char b = 0b0111;

    printf("a=0x%x, b=0x%x\na|b = 0x%x\n", a, b, a|b);
}
```

- Assembly

```
fmt: .string "a=0x%x, b=0x%x\na|b = 0x%x\n"
    .balign 4
    .global main

main:
    stp  x29, x30, [sp, -16]!
    mov  x29, sp
    ldr  x0, =fmt
    mov  x1, 0b1001
    mov  x2, 0b0111
    orr  x3, x2, x1
    bl printf

    ldp x29, x30, [sp], 16
    ret
```

Coding practice —Bitwise XOR

- C

```c
#include <stdio.h>

void main(int argc, char *argv[]) {
    char a = 0b1001;
    char b = 0b0111;

    printf("a=0x%x, b=0x%x\na&b = 0x%x\n", a, b, a^b);
}
```

- Assembly

```
fmt: .string "a=0x%x, b=0x%x\na&b = 0x%x\n"
    .balign 4
    .global main

main:
    stp  x29, x30, [sp, -16]!
    mov  x29, sp
    ldr  x0, =fmt
    mov  x1, 0b1001
    mov  x2, 0b0111
    eor  x3, x2, x1
    bl printf

    ldp x29, x30, [sp], 16
    ret
```

Exercise

- write an assembly program to check a given number is odd or even

(tips: simple test would be to check the least significant bit of the number. If this is 1, the number is odd, else the number is even.)

References

- course lecture slides
- https://db.tt/w6N4iGtPBX
- https://www.tutorialspoint.com/assembly_programming/assembly_logical_instructions.htm