Arhitektura računara

Rukovanje bitima

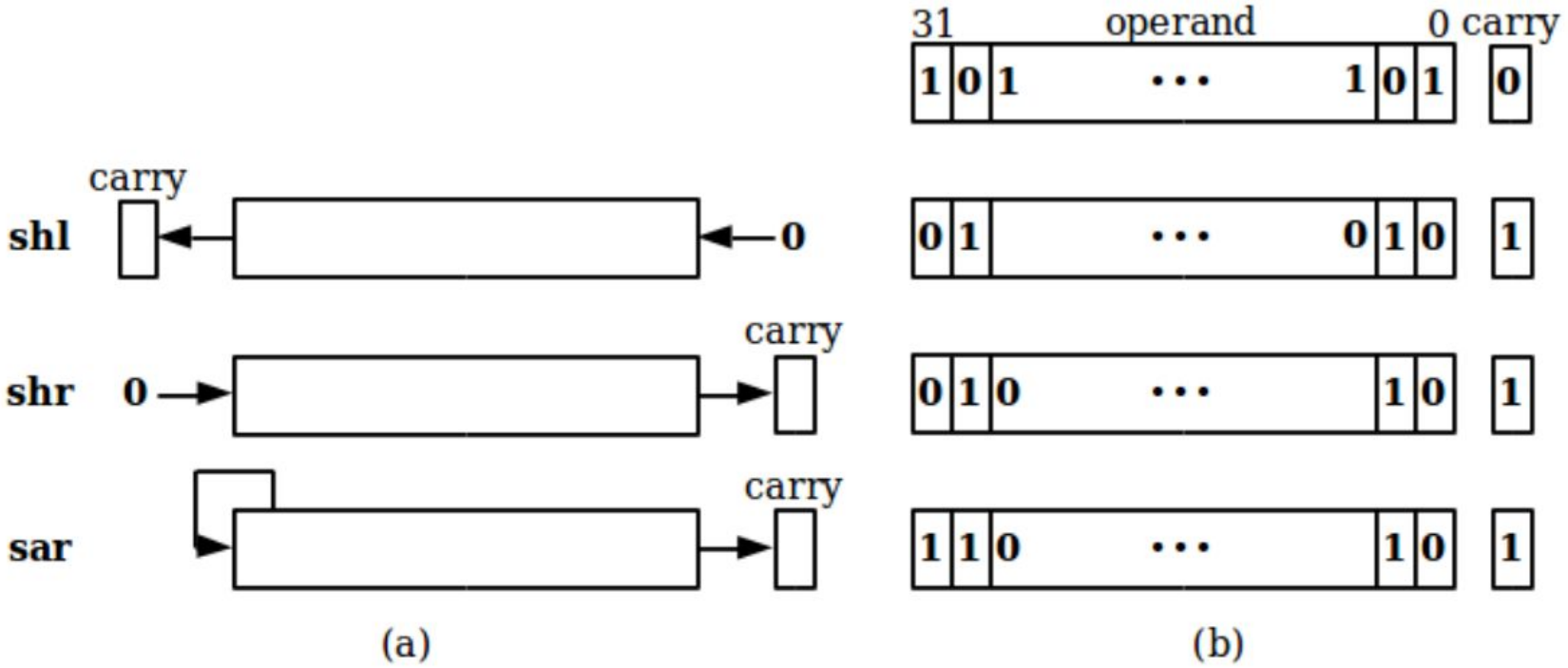
Osnovne logičke naredbe

- **and** - binarno I
- **or** - binarno ILI
- **xor** - binarno EKSKLUZIVNO ILI
- **not** - binarno NE
- **neg** - aritmetička negacija
- **test** - binarno I (bez promene odredišta)

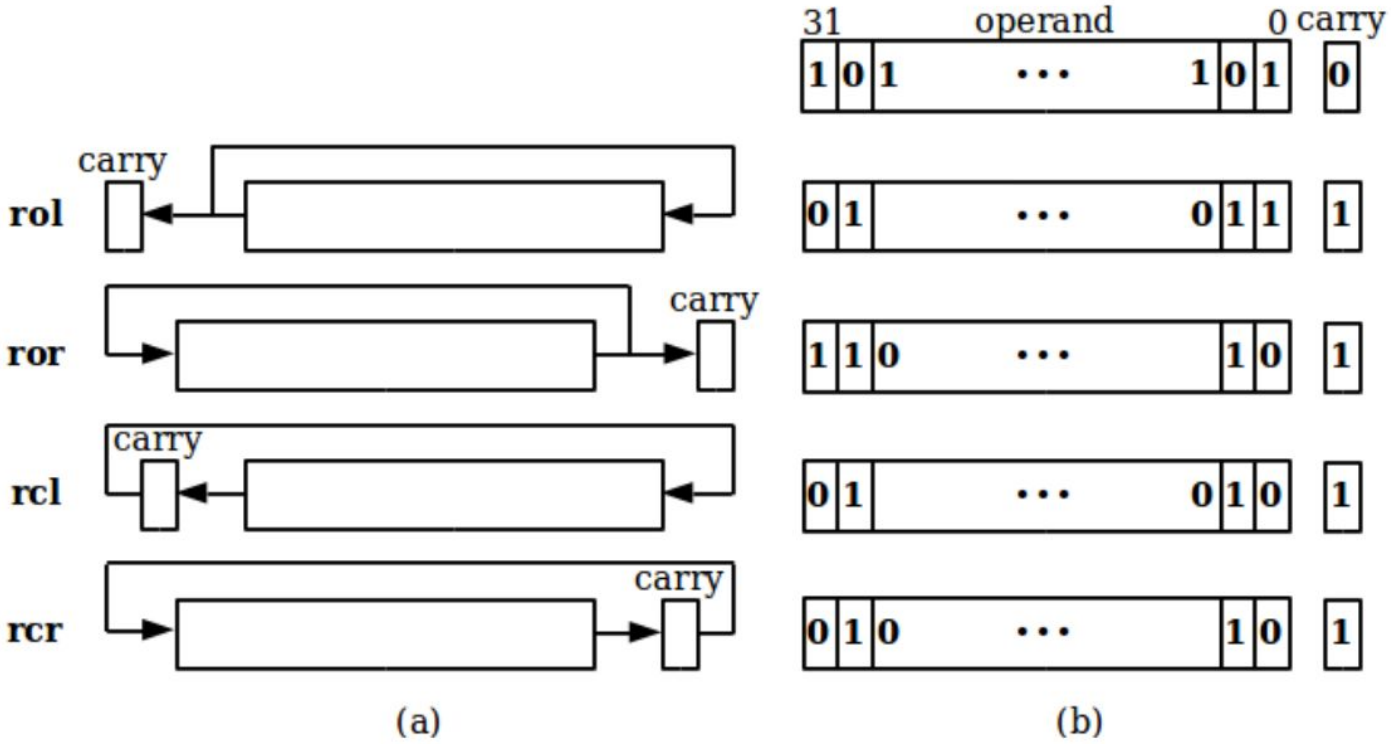
Maskiranje

- Postavljanje najnižeg bita registra eax na 0
  - maska: **`0b11111111111111111111111111110 = 0xfffffe`**
  - **`andl $0xfffffe, %eax`**
- Postavljanje najnižeg bita registra eax na 1
  - maska: **`0b00000000000000000000000000001 = 1`**
  - **`orl $1, %eax`**
- Invertovanje najnižeg bita registra eax
  - maska: **`0b00000000000000000000000000001 = 1`**
  - **`xorl $1, %eax`**

Naredbe za binarno pomeranje



Naredbe za binarno pomeranje



Naredbe za binarno pomeranje

- Naredbe za binarno pomeranje primaju dva operanda.
- Drugi, odredišni operand je ili registar, ili memorijska lokacija.
- Prvi operand označava za koliko binarnih mesta se vrši pomeranje i može biti:
  - neposredni operand, ili
  - registar **%cl** (koristi se **cl** bez obzira na sufiks instrukcije i veličinu drugog operanda)

Primeri primene

```
cmpl $0, %eax
je …
```

```
andl %eax, %eax
jz …
```

```
movl $0, %eax
```

```
xorl %eax, %eax
```

```
movl $2, %edx
mull %edx
```

```
shll $1, %eax
```

```
shrl $1, %eax
```

```
movl $2, %ecx
movl $0, %edx
divl %ecx
```

Prebrojavanje jedinica u binarnom broju

```
int a = 12345;
int j = 0;
while(a != 0) {
    if(a & 1 !=
0)
        j++;
    a >>= 1;
}
```

```
    movl $12345, %eax
    movl $0, %ecx
petlja:
    cmpl $0, %eax
    je kraj
    testl $1, %eax
    jz nula
    incl %ecx
nula:
    shrl $1, %eax
    jmp petlja
kraj:
    movl $1, %eax
    movl $0, %ebx
    int $0x80
```

Kontrolna suma i paritet

- Upotreba kontrolnih brojeva/suma
  - provera autentičnosti/ispravnosti podataka
  - korekcija neispravnosti podataka
- Primeri:
  - CRC, MD5, SHA, ...

Paritet

- Parni i neparni
- **Horizontalni paritet**
  1. 00101001 **1**
  2. 01010101 **0**
  3. 01010100 **1**
  4. 01100101 **0**
  5. 00011111 **1**
- **Vertikalni paritet**
  1. 00101001
  2. 01010101
  3. 01010100
  4. 01100101
  5. 00011111

  ∑ **01010010**

Primer: Vertikalni paritet za 32-bitni niz

```
br_elem = 6
niz:
    .long 0b10101010100101010001010100111111
    .long 0b10101010111011111010101010001010
    .long 0b11111111110000000001111111000000
    .long 0b11101001010110100101101010101010
    .long 0b00010101010100101010101010100101
    .long 0b11000101001010001001000100101010
checksum: .long 0
…
    xorl %eax, %eax
    movl $1, %edx
sledeci_bit:
    xorl %ecx, %ecx
    xorl %esi, %esi
sledeci_el:
    movl niz(,%esi,4), %ebx
    andl %edx, %ebx
    jz bit_nula
    incl %ecx
```

```
bit_nula:
    incl %esi
    cmpl $br_elem, %esi
    jl sledeci_el
    shrl $1, %ecx
    rcrl $1, %eax
    shll $1, %edx
    jnc sledeci_bit
    movl %eax, checksum
kraj:
    movl $1, %eax
    movl $0, %ebx
    int $0x80
```