Programowanie niskopoziomowe

Laboratorium 04 – Materiały uzupełniające

„Instrukcje arytmetyczne i logiczne”

INC, DEC

- **Dodaje 1, odejmuje 1 od operandu celu**
  - **Operand może być rejestrem lub pamięcią**
- **INC cel**
  - **Destination ← Deestination + 1**
- **DEC cel**
  - **Destination ← Deestination - 1**

ADD, SUB

- ADD cel, źródło
  - destination ← destination + source
- SUB cel, source
  - Zmienia znak operandu. Operand może być rejestrem lub pamięcią.
  - destination – source
- Reguły dotyczące operandów jak dla instrukcji MOV

NEG (negate)

- NEG destination
- Zmienia znak operandu. Operand może być rejestrem lub pamięcią.

MUL

Instrukcja MUL (unsigned multiply) mnoży 8-, 16-, or 32-bitowy operand odpowiednio przez AL, AX, or EAX.

Składnia:

```
MUL r/m8
MUL r/m16
MUL r/m32
```

Table 7-2 MUL Operands.

Multiplicand	Multiplier	Product
AL	reg/mem8	AX
AX	reg/mem16	DX:AX
EAX	reg/mem32	EDX:EAX

IMUL

Mnożenie ze znakiem, dla jednego operandu zachowuje się jak MUL, dopuszczalna inna składnia:

```
IMUL     source
IMUL     source,immed          (286+)
IMUL     destination, source,immed8   (286+)
IMUL     destination, source (386+)
```

DIV

Instrukcja DIV (unsigned divide) wykonuje dzielenie 8-, 16-, 32-bitowe na liczbach bez znaku

Jeden operand jest wymagany (rejestr lub pamięć), który określa dzielnik

Składnia:

```
DIV reg/mem8
DIV reg/mem16
DIV reg/mem32
```

Domyślne operandy:

Dividend	Divisor	Quotient	Remainder
AX	*r/m8*	AL	AH
DX:AX	*r/m16*	AX	DX
EDX:EAX	*r/m32*	EAX	EDX

IDIV

Dzielenie ze znakiem, składnia jak dla DIV:

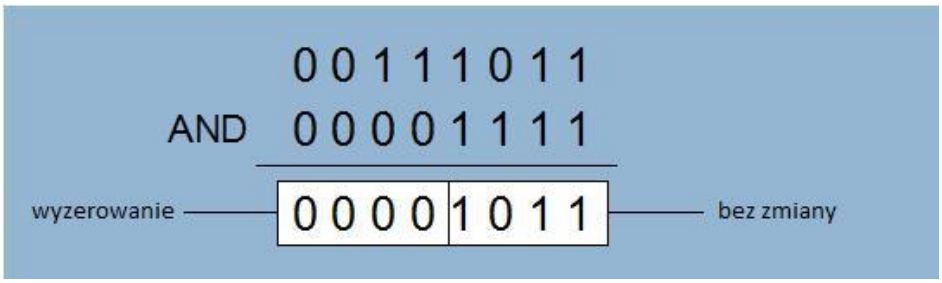
AND (Iloczyn logiczny)

- Składnia

  and destination,source

- Przykładowe zastosowanie

  - Wyzerowanie jednego lub więcej bitów (byte, word, or doubleword)
  - Izolowanie jednego lub więcej bitów (byte, word, or doubleword)



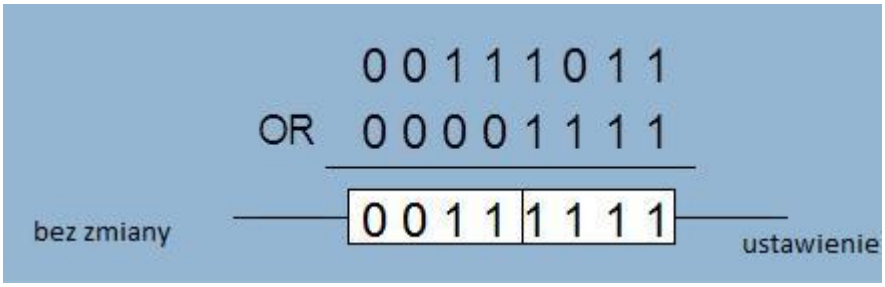
OR (Suma logiczna)

- Składnia

  or destination,source

- Przykładowe zastosowanie

  - Ustawienie jednego lub więcej bitów (byte, word, or doubleword)



XOR

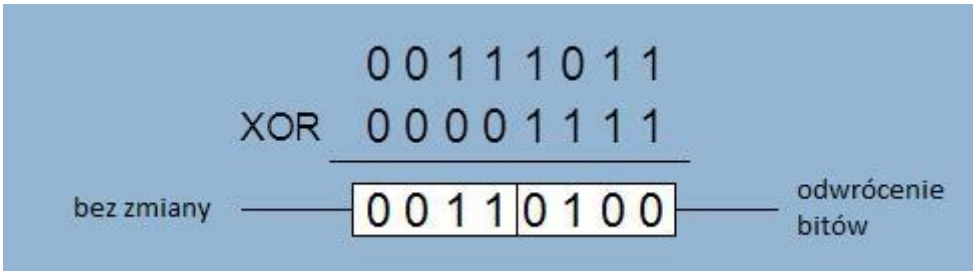
- Składnia

  xor destination,source

- Przykładowe zastosowanie

  - Przełączanie jednego lub więcej bitów (byte, word, or doubleword)
  - Wyzerowanie rejestru

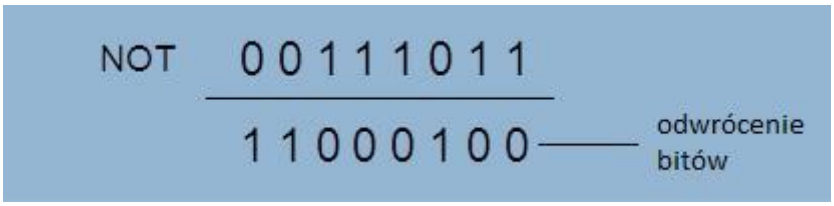
  xor AX,AX



NOT (negacja logiczna)

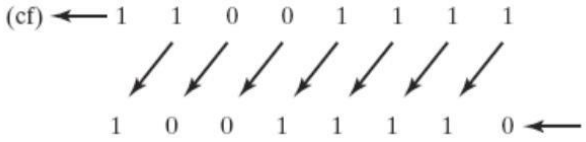
- Składnia

  not destination



SHL (shift left)

Rozkaz SHL (shift left) wykonuje logiczne przesunięcie w lewo, wypełniając najmniej znaczący bit wartością 0.



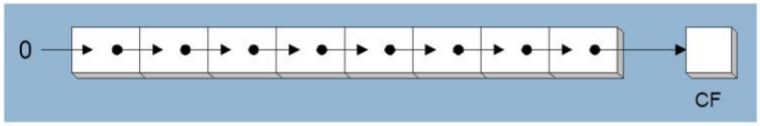
Typy operandów dla SHL:

```
SHL reg,imm8
SHL mem,imm8
SHL reg,CL
SHL mem,CL
```

(Tak samo dla instrukcji przesunięcia i obrotów)

SHR (shift right)

Instrukcja SHR (shift right) wykonuje logiczne przesunięcie w prawo. Najstarszy bit jest wypełniany wartością 0.



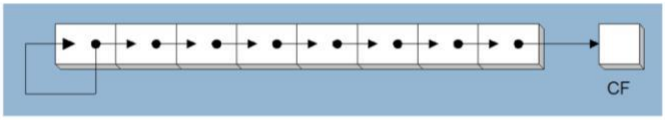
Przesunięcie w prawo o *n* bitów dzieli operand przez 2^n

```
mov dl,80
shr dl,1                ; DL = 40
shr dl,2                ; DL = 10
```

SAL (shift arithmetic left), SAR (shift arithmetic right)

SAL (shift arithmetic left) robi to samo co SHL.

SAR (shift arithmetic right) wykonuje przesunięcie prawo z zachowaniem wartości najstarszego bitu.



Przesunięcie arytmetyczne zachowuje znak liczby

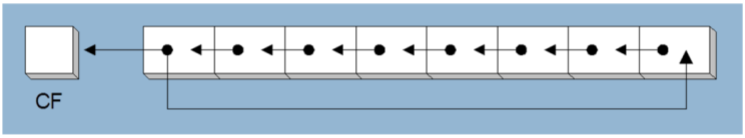
```
mov dl,-80
sar dl,1                    ; DL = -40
sar dl,2                    ; DL = -10
```

ROL ROR

ROL (rotate) przesuwa każdy bit w lewo

Najstarszy bit jest kopiowany zarówno do flagi przeniesienia (Carry flag) i do najmłodszego bitu

Żaden bit nie jest tracony



ROR (rotate right) przesuwa każdy bit w prawo

Najmłodszy bit jest kopiowany zarówno do flagi przeniesienia (Carry flag) i do najstarszego bitu

Żaden bit nie jest tracony