6. Celočíselné instrukce – pokračování

Jiné instrukce

- LEA
- NOP
- XLAT/XLATB
- UD2
- CPUID

LEA dest,src (Load Effective Address)

8D /r LEA r16,m (LEA r32,m)

Naplní cílový registr efektivní adresou zdrojového operandu.
Příznaky nemění.

Příklad použití:

```
lea bx,[bp+si+6]
```

NOP (No Operation)

90 NOP

Prázdná operace – provede se instrukce XCHG (E)AX,(E)AX.

XLAT src (Table Look-up Translation)
XLATB

D7 XLAT m8 XLATB

[DS:(E)BX + ZeroExtend(AL)] → AL

Explicitní vyjádření operandu v instrukci XLAT umožňuje přepis segmentového registru, bázovým registrem zdrojového operandu je vždy registr (E)BX.
Příznaky nemění.

UD2 (Undefined Instruction, Pentium® Pro)

OF OB UD2

Instrukce s „neplatným“ operačním kódem. Používá se pro účely testování.
Příznaky nemění.

CPUID (CPU Identification, Pentium®)

0F A2 CPUID

Initial EAX Value	Value Information Provided about the Processor
0	EAX Maximum CPUID Input Value (2 for the Pentium ® Pro processor processor and 1 for the Pentium processor) EBX "Genu" ECX "ntel" EDX "inel"
1	EAX Version Information (Type, Family, Model, and Stepping ID) EBX Reserved ECX Reserved EDX Feature Information
2	EAX Cache and TLB Information EBX Cache and TLB Information ECX Cache and TLB Information EDX Cache and TLB Information

Instrukce pro práci s bity a slabikami

- BT
- BTS
- BTR
- BTC
- BSF
- BSR
- SETcc

BT, BTS, BTR, BTC

BT dest,src (Bit Test, 386)

0F A3	BT r/m16,r16	(BT r/m32,r32)
0F BA /4 i	BT r/m16,imm8	(BT r/m32,imm8)

Nastaví CF na hodnotu bitu cílového operandu, jehož offset je dán zdrojovým operandem.
Hodnoty příznaků OF, SF, ZF, AF a PF nejsou definovány.
Je-li cílovým operandem registr, pak původní hodnota offsetu je upravena podle vztahu (offset mod 16, resp. offset mod 32).
Je-li cílovým operandem paměť a je-li offset dán přímým operandem, pak offset může nabývat hodnot v rozsahu <0, 31>.
Je-li cílovým operandem paměť a je-li offset dán obsahem registru, pak původní hodnota offsetu není nijak upravována (může se pohybovat v intervalu $<-2^{31},2^{31}-1>$).

Offset bitu v registru se počítá zprava doleva počínaje nulou.

Offset bitu v paměti se počítá zprava doleva počínaje nulou v adresované slabice a pro kladné hodnoty pokračuje postupně (vždy opět zprava doleva) ve slabikách následujících, tj. ve slabikách s postupně rostoucími adresami – pro záporné hodnoty ofsetu se postupuje v opačném směru.

BTS dest,src (Bit Test and Set, 386)

0F AB	BTS r/m16,r16	(BTS r/m32,r32)
0F BA /5 i	BTS r/m16,imm8	(BTS r/m32,imm8)

Nastaví CF na hodnotu bitu cílového operandu, jehož offset je dán zdrojovým operandem a nastaví tento bit na hodnotu 1.
Hodnoty příznaků OF, SF, ZF, AF a PF nejsou definovány.
Je-li cílovým operandem registr, pak původní hodnota offsetu je upravena podle vztahu (offset mod 16, resp. offset mod 32).
Je-li cílovým operandem paměť a je-li offset dán přímým operandem, pak offset může nabývat hodnot v rozsahu <0, 31>.
Je-li cílovým operandem paměť a je-li offset dán obsahem registru, pak původní hodnota offsetu není nijak upravována (může se pohybovat v intervalu $<-2^{31},2^{31}-1>$).

BTR dest,src (Bit Test and Reset, 386)

0F B3	BTR r/m16,r16	(BTR r/m32,r32)
0F BA /6 i	BTR r/m16,imm8	(BTR r/m32,imm8)

Nastaví CF na hodnotu bitu cílového operandu, jehož offset je dán zdrojovým operandem a tento bit vynuluje.
Hodnoty příznaků OF, SF, ZF, AF a PF nejsou definovány.
Je-li cílovým operandem registr, pak původní hodnota offsetu je upravena podle vztahu (offset mod 16, resp. offset mod 32).
Je-li cílovým operandem paměť a je-li offset dán přímým operandem, pak offset může nabývat hodnot v rozsahu <0, 31>.
Je-li cílovým operandem paměť a je-li offset dán obsahem registru, pak původní hodnota offsetu není nijak upravována (může se pohybovat v intervalu $<-2^{31},2^{31}-1>$).

BTC dest,src (Bit Test and Complement, 386)

0F BB	BTC r/m16,r16	(BTC r/m32,r32)
0F BA /7 i	BTC r/m16,imm8	(BTC r/m32,imm8)

Nastaví CF na hodnotu bitu cílového operandu, jehož offset je dán zdrojovým operandem a změní hodnotu tohoto bitu.
Hodnoty příznaků OF, SF, ZF, AF a PF nejsou definovány.
Je-li cílovým operandem registr, pak původní hodnota offsetu je upravena podle vztahu (offset mod 16, resp. offset mod 32).
Je-li cílovým operandem paměť a je-li offset dán přímým operandem, pak offset může nabývat hodnot v rozsahu <0, 31>.
Je-li cílovým operandem paměť a je-li offset dán obsahem registru, pak původní hodnota offsetu není nijak upravována (může se pohybovat v intervalu $<-2^{31},2^{31}-1>$).

BSF, BSR

BSF dest,src (Bit Scan Forward, 386)

0F BC BSF r16,r/m16 (BSF r32,r/m32)

Hledá první nenulový bit zprava ve zdrojovém operandu (prohlíží maximálně 16, resp. 32 bitů). Pokud takový bit nalezne, vynuluje příznak ZF (0 → ZF) a offset tohoto bitu uloží do cílového operandu. V opačném případě pouze nastaví příznak ZF (1 → ZF). Hodnoty příznaků CF,OF, SF, AF a PF nejsou definovány.

BSR dest,src (Bit Scan Reverse, 386)

0F BD BSR r16,r/m16 (BSR r32,r/m32)

Hledá první nenulový bit zleva ve zdrojovém operandu (prohlíží maximálně 16, resp. 32 bitů). Pokud takový bit nalezne, vynuluje příznak ZF (0 $\rightarrow$ ZF) a offset tohoto bitu uloží do cílového operandu. V opačném případě pouze nastaví příznak ZF (1 $\rightarrow$ ZF). Hodnoty příznaků CF,OF, SF, AF a PF nejsou definovány.

SETcc

SETcc dest (Set Byte of Condition, 386)

0F 97 /r	SETA r/m8	set byte if above (CF=0 and ZF=0)
0F 93 /r	SETAE r/m8	set byte if above or equal (CF=0)
0F 92 /r	SETB r/m8	set byte if below (CF=1)
0F 96 /r	SETBE r/m8	set byte if below or equal (CF=1 or ZF=1)
0F 92 /r	SETC r/m8	set byte if carry (CF=1)
0F 94 /r	SETE r/m8	set byte if equal (ZF=1)
0F 9F /r	SETG r/m8	set byte if greater (ZF=0 and SF=OF)
0F 9D /r	SETGE r/m8	set byte if greater or equal (SF=OF)
0F 9C /r	SETL r/m8	set byte if less (SF<>OF)
0F 9E /r	SETLE r/m8	set byte if less or equal (ZF=1 or SF<>OF)
0F 96 /r	SETNA r/m8	set byte if not above (CF=1 or ZF=1)
0F 92 /r	SETNAE r/m8	set byte if not above or equal (CF=1)
0F 93 /r	SETNB r/m8	set byte if not below (CF=0)
0F 97 /r	SETNBE r/m8	set byte if not below or equal (CF=0 and ZF=0)
0F 93 /r	SETNC r/m8	set byte if not carry (CF=0)

0F 95 /r	SETNE r/m8	set byte if not equal (ZF=0)
0F 9E /r	SETNG r/m8	set byte if not greater (ZF=1 or SF<>OF)
0F 9C /r	SETNGE r/m8	set byte if not greater or equal (SF<>OF)
0F 9D /r	SETNL r/m8	set byte if not less (SF=OF)
0F 9F /r	SETNLE r/m8	set byte if not less or equal (ZF=0 and SF=OF)
0F 91 /r	SETNO r/m8	set byte if not overflow (OF=0)
0F 9B /r	SETNP r/m8	set byte if not parity (PF=0)
0F 99 /r	SETNS r/m8	set byte if not sign (SF=0)
0F 95 /r	SETNZ r/m8	set byte if not zero (ZF=0)
0F 90 /r	SETO r/m8	set byte if overflow (OF=1)
0F 9A /r	SETP r/m8	set byte if parity (PF=1)
0F 9A /r	SETPE r/m8	set byte if parity even (PF=1)
0F 9B /r	SETPO r/m8	set byte if parity odd (PF=0)
0F 98 /r	SETS r/m8	set byte if sign (SF=1)
0F 94 /r	SETZ r/m8	set byte if zero (ZF=1)

If podmínka then 1 → r/m8 else 0 → r/m8. Příznaky nemění.

Řetězové instrukce

Předpony:

REP/REPE/REPZ/REPNE/REPNZ

Instrukce:

- **MOVS/MOVSB/MOVSW**/MOVSD
- **CMPS/CMPSB/CMPSW**/CMPSD
- **SCAS/SCASB/SCASW**/SCASD
- **LODS/LODSB/LODSW**/LODSD
- **STOS/STOSB/STOSW**/STOSD
- INS/INSB/INSW/INSD
- OUTS/OUTSB/OUTSW/OUTSD

Pozn.: Základní řetězové instrukce (označené modře) NASM na rozdíl od jiných asemblerů nezná !!!

REP/REPE/REPZ/REPNE/REPNZ

REP/REPE/REPZ/REPNE/REPNZ (Repeat String Operation Prefix)

F3 REP/REPE/REPZ
F2 REPNE/REPNZ

1) dokud (E)CX <> 0 opakuj
{MOVS/LODS/STOS/INS/OUTS}, dec (E)CX;

2) if (E)CX <> 0 then
opakuj
{CMPS/SCAS}, dec (E)CX;
dokud not ((E)CX = 0) or
(REPE/REPZ and (ZF=0)) or
(REPNE/REPNZ and (ZF=1)));

Předpona příznaky nemění.

MOVS/MOVSB/MOVSW/MOVSD

MOVS dest,src (Move Data from String to String)

MOVSB/MOVSW/MOVSD

A4	MOVS m8, m8	MOVSB
A5	MOVS m16, m16	MOVSW
A5	MOVS m32, m32	MOVSD

Přenese slabiku/slovo/dvouslovo z adresy DS:(E)SI na adresu ES:(E)DI a v závislosti na hodnotě příznaku DF (0/1) zvýší/sníží obsahy obou indexregistrů o hodnotu 1/2/4 (pro slabiku/slovo/dvouslovo).
Explicitní vyjádření operandů v instrukci MOVS standardně umožňuje přepis segmentového registru zdrojového operandu - adresa cílového operandu je vždy dána předpisem ES:(E)DI, indexregistrem zdrojového operandu je vždy registr (E)SI. V NASM se případný prefix pro přepis segmentového registru zdrojového operandu píše přímo před implicitní instrukce (MOVSB/MOVSW/MOVSD).
Příznaky nemění.

Příklady použití instrukcí MOVSB a MOVSW:

```
        …
                mov cx, 100             ; nastavení počtu přenášených položek
        cld                             ; nastavení směru přenosu
                                        ; (od počátku ke konci paměti)
rep     movsw                           ; přenos 100 slov, [ds:si] → [es:di]
        …                               ; oba indexregistry postupně
                                ; zvyšují své hodnoty o dvojku
```

```
        …
                mov cx, 50              ; nastavení počtu přenášených položek
        std                             ; nastavení směru přenosu
                                        ; (od konce k počátku paměti)
        es                              ; prefix pro změnu zdrojového
                                ; segmentového registru
rep     movsb                           ; přenos 50 slabik, [es:si] → [es:di]
        …                               ; oba indexregistry postupně
                                ; snižují své hodnoty o jedničku
```

CMPS/CMPSB/CMPSW/CMPSD

CMPS dest,src (Compare String Operands)

CMPSB/CMPSW/CMPSD

A6	CMPS m8, m8	CMPSB
A7	CMPS m16, m16	CMPSW
A7	CMPS m32, m32	CMPSD

Porovná slabiky/slova/dvouslova na adresách DS:(E)SI a ES:(E)DI (nastaví příznaky na základě výsledku **src – dest !!!**) a v závislosti na hodnotě příznaku DF (0/1) zvýší/sníží obsahy obou indexregistrů o hodnotu 1/2/4 (pro slabiku/slovo/dvouslovo).
Explicitní vyjádření operandů v instrukci CMPS umožňuje přepis segmentového registru zdrojového operandu - adresa cílového operandu je vždy dána předpisem ES:(E)DI, indexregistrem zdrojového operandu je vždy registr (E)SI. V NASM se případný prefix pro přepis segmentového registru zdrojového operandu píše přímo před implicitní instrukce (MOVSB/MOVSW/MOVSD).

Příklad použití instrukce CMPSB (použití instrukce CMPSW by bylo podobné):

```
        …
        mov cx, 100     ; nastavení počtu porovnávaných položek
        cld             ; nastavení směru porovnávání
  repe  cmpsb           ; porovnávání (max 100) slabik, [ds:si] ? [es:di]
                        ; oba indexregistry postupně
                        ; zvyšují své hodnoty o jedničku
        je stejne
ruzne:  …
        …
        jmp pokracuj
stejne: …
        …
pokracuj:
        …
```

SCAS/SCASB/SCASW/SCASD

SCAS dest (Scan String)

SCASB/SCASW/SCASD

AE	SCAS m8	SCASB
AF	SCAS m16	SCASW
AF	SCAS m32	SCASD

Porovná slabiku/slovo/dvouslovo na adrese ES:(E)DI s obsahem střádače AL/AX/EAX (nastaví příznaky na základě výsledku **a – dest !**) a v závislosti na hodnotě příznaku DF (0/1) zvýší/sníží obsah indexregistru (E)DI o hodnotu 1/2/4 (pro slabiku/slovo/dvouslovo). Explicitní vyjádření operandu v instrukci SCAS nemá prakticky žádný význam - adresa cílového operandu je vždy dána předpisem ES:(E)DI.

Příklad použití instrukce SCASW (použití instrukce SCASB by bylo podobné):

```
            …
            mov cx, 200       ; nastavení počtu prohledávaných položek
            cld               ; nastavení směru prohledávání
            mov ax,1000       ; bude se hledat číslo 1000
    repne   scasw             ; prohledávání (max 200) slov, ax ? [es:di]
                              ; indexregistr di zvyšuje svou hodnotu o dvojku
            jne nenasel
nasel:      sub di,2          ; indexregistr di ukazuje na nalezené číslo
            …
            jmp pokracuj
nenasel:    …
            …
pokracuj:
            …
```

LODS/**LODSB**/**LODSW**/LODSD

STOS/**STOSB**/**STOSW**/STOSD

LODS src (Load String)

LODSB/LODSW/LODSD

AC	LODS m8	LODSB
AD	LODS m16	LODSW
AD	LODS m32	LODSD

Načte slabiku/slovo/dvouslovo z adresy DS:(E)SI do střádače AL/AX/EAX a v závislosti na hodnotě příznaku DF (0/1) zvýší/sníží obsah indexregistru (E)SI o hodnotu 1/2/4 (pro slabiku/slovo/dvouslovo). Explicitní vyjádření operandu v instrukci LODS umožňuje přepis segmentového registru, indexregistrem zdrojového operandu je vždy registr (E)SI.
Příznaky nemění.

STOS dest (Store String)

STOSB/STOSW/STOSD

AA	STOS m8	STOSB
AB	STOS m16	STOSW
AB	STOS m32	STOSD

Uloží obsah střádače AL/AX/EAX na adresu ES:(E)DI a v závislosti na hodnotě příznaku DF (0/1) zvýší/sníží obsah indexregistru (E)DI o hodnotu 1/2/4 (pro slabiku/slovo/dvouslovo).
Explicitní vyjádření operandu v instrukci STOS nemá prakticky žádný význam - adresa cílového operandu je vždy dána předpisem ES:(E)DI.
Příznaky nemění.

Příklad použití instrukcí LODSW a STOSW (použití instrukcí LODSB a STOSB je podobné) - změna znamének čísel v poli 500 čísel Integer:

```
        …
        mov cx, 500     ; nastavení počtu čísel v poli
        lds si,pole     ; ukazatel na pole do ds:si
        push ds
        pop es          ; kopie adresy datového segmentu z ds do es
        mov di,si       ; kopie offsetu z registru si do registru di
        cld             ; nastavení směru procházení polem
cykl:
        lodsw           ; číslo z pole do ax
        neg ax          ; změna znaménka čísla v ax
        stosw           ; číslo z ax na původní místo v poli
        loop cykl
        …
```

Pozn.: Uvedený program nezmění znaménko čísla -32768 !!!

INS/INSB/INSW/INSD

INS dest (Input from Port to String)

INSB/INSW/INSD

6C	INS m8, DX	INSB
6D	INS m16, DX	INSW
6D	INS m32, DX	INSD

Přenese slabiku/slovo/dvouslovo z I/O portu adresovaného obsahem registru DX na adresu ES:(E)DI a v závislosti na hodnotě příznaku DF (0/1) zvýší/sníží obsah indexregistru (E)DI o hodnotu 1/2/4 (pro slabiku/slovo/dvouslovo).
Explicitní vyjádření operandu v instrukci INS nemá prakticky žádný význam - adresa cílového operandu je vždy dána předpisem ES:(E)DI.
Příznaky nemění.

OUTS/OUTSB/OUTSW/OUTSD

OUTS scr (Output String to Port)

OUTSB/OUTSW/OUTSD

6E	OUTS DX, m8	OUTSB
6F	OUTS DX, m16	OUTSW
6F	OUTS DX, m32	OUTSD

Přenese slabiku/slovo/dvouslovo z adresy DS:(E)SI na I/O port adresovaný obsahem registru DX a v závislosti na hodnotě příznaku DF (0/1) zvýší/sníží obsah indexregistru (E)SI o hodnotu 1/2/4 (pro slabiku/slovo/dvouslovo).
Explicitní vyjádření operandu v instrukci OUTS umožňuje přepis segmentového registru, indexregistrem zdrojového operandu je vždy registr (E)SI.
Příznaky nemění.

Instrukce dekadické aritmetiky

- DAA
- DAS
- AAA
- AAS
- AAM
- AAD

DAA, DAS

DAA (Decimal Adjust AL after Addition)

27 DAA

if ((AL and 0FH) > 9) or (AF = 1) then begin
    AL + 6 → AL
    1 → AF
    (CF or *carry from* AL + 6) → CF
end
else 0 → AF
if ((AL and F0H) > 90H) or (CF = 1) then begin
    AL + 60H → AL
    1 → CF
end
else 0 → CF

Dále nastavuje příznaky SF, ZF a PF (podle výsledku), hodnota příznaku OF není definována.

DAS (Decimal Adjust AL after Subtraction)

2F DAS

if ((AL and 0FH) > 9) or (AF = 1) then begin
    AL - 6 → AL
    1 → AF
    (CF or *borrow from* AL - 6) → CF
end
else 0 → AF
if ((AL and F0H) > 90H) or (CF = 1) then begin
    AL - 60H → AL
    1 → CF
end
else 0 → CF

Dále nastavuje příznaky SF, ZF a PF (podle výsledku)), hodnota příznaku OF není definována.

AAA, AAS, AAM, AAD

AAA (ASCII Adjust after Addition)

37 AAA

if ((AL and 0FH) > 9) or (AF = 1) then begin
    AL + 6 → AL
    AH + 1 → AH
    1 → AF
    1 → CF
end
else begin
    0 → AF
    0 → CF
end
((AL and 0FH) → AL

Hodnoty příznaků OF, SF, ZF a PF nejsou definovány.

AAS (ASCII Adjust after Subtraction)

3F AAS

if ((AL and 0FH) > 9) or (AF = 1) then begin
    AL - 6 → AL
    AH - 1 → AH
    1 → AF
    1 → CF
end
else begin
    0 → AF
    0 → CF
end
((AL and 0FH) → AL

Hodnoty příznaků OF, SF, ZF a PF nejsou definovány.

AAM (ASCII Adjust after Multiplication)

D4 0A AAM

AL div 10 → AH
AL mod 10 → AL

Nastavuje příznaky SF, ZF a PF (podle výsledku), hodnoty příznaků OF, CF a AF nejsou definovány.

AAD (ASCII Adjust before Division)

D5 0A AAD

$AL + AH * 10 \rightarrow AL$
$0 \rightarrow AH$

Nastavuje příznaky SF, ZF a PF (podle výsledku)), hodnoty příznaků OF, CF a AF nejsou definovány.