Aritmetikai utasítások II.

A korábbiakban látott utasításokon túl további aritmetikai utasítások kerülnek bemutatásra. Az egyes utasítások után példákat is nézünk azok működésére.

További aritmetikai utasítások

`ADC:` A többszavas összeadást támogatja. Előfordulhat, hogy az összeadás két tagja nem fér el 16 biten, akkor a két operandust a (DX:AX) és a (CX:BX) regiszterpárban tárolhatjuk. Az összeadás műveletét előjeltől függetlenül a

```
ADD AX, BX
ADC DX, CX
```

utasításokkal kell elvégezni. Az `ADC` utasítás is hatással van a `C`, `O` és az `S` flagre. A különbség az `ADD` és az `ADC` utasítás között az, hogy az `ADC` utasítás figyelembe veszi a Carry értékét is. Az eredmény DX:AX-ben keletkezik. Az `ADC` utasítás a DX és CX regiszter tartalmának összeadása után ismét beállítja a Carry értékét aszerint, hogy lett-e túlcsordulás az összeg után.

`CWD:` Az AX értékét DX:AX-re állítja be előjelhelyesen.

Példák:

Legyen az (DX:AX)-ben tárolt érték: 0003 0020h, a (CX:BX)-ben tárolt érték: 0034 0FFFh

```
ADD AX, BX          ;AX=101Fh      O=1, C=0, S=1
ADC DX, CX          ;DX=0037       O=0, C=0, S=0
```

```
ADD AX, BX          ;AX=101Fh      O=1, C=0, S=1
```

Nem ad helyes eredményt, ha az operandus (DX:AX)-ben van!

```
ADC DX, CX           ;DX:AX=0037 0020h O=0, C=0, S=0
```

Nem ad helyes eredményt, ha az operandus (DX:AX)-ben van!

`INC:` Az operandusában megadott értéket 1-gyel növeli. Nincs hatással az átvitelre.

`AAA:` A pakolatlan (ASCII kódú, egy számjegy nyolc biten) decimális számok összeadását támogatja.

Példa: Végezzük el az alábbi összeadást és hozzuk pakolatlan BCD kódra az eredményt! A decimális számok ASCII kódjai 30H - 39H között vannak. `AX = 532h, BX = 38h`, vagyis decimálisan szerint AX = 52d, BX=8d

```
ADD AL, BL          ;AH=05, AL=6Ah, A=0
AAA                 ;AL alsó 4 bitje A, és A > 9,
                    ;vagyis AL=AL+6, AH=AH+1,
                    ;tehát AX=0600h = 60d
```

`DAA:` A pakolt (ASCII kódú, egy számjegy 4 biten) decimális számok összeadását támogatja. Az `A` flag jelzi, ha a művelet során a 3. bitről túlcsordulás történik, ilyenkor `A=1`.

`SUB:` két operandusú kivonás művelet, az első operandusból vonja ki a második operandus értékét. Az eredmény az első operandusban keletkezik.

Példa: Végezzük el a műveletet: 40h - 27h!

```
MOV AX, 40h         AX=0040h
SUB AX, 27h         AX=0019h        S=0, O=0, C=0
```

Végezzük el a műveletet: 40h - 57h!

```
MOV AX, 40h         AX=0040h
SUB AX, 57h         AX=FFE9h        S=1, O=1, C=0
```

`SBB:` A többszavas kivonás műveletét segíti.

`DEC:` Az operandusának az értékét 1-gyel csökkenti. Nincs hatással a Carry-re.

`AAS:` Két ASCII szám kivonása után korrigál:

1. Ha `AL` alsó 4 bitje ≤ 9 és az `A` flag 0, akkor a 3. lépés jön.
2. Ha `AL` = `AL`-6, `AH` = `AH`-1, `A` = 1

3. AL felső 4 bitje 0

4. C = A

DAS: Két pakolt ASCII szám kivonása után korrigál:

1. Ha A = 0 vagy az AL alsó 4 bitje ≤ 9, akkor AL = AL-6, A = 1.

2. Ha C = 1 vagy AL felső 4 bitje ≤ 9, akkor AL = AL - 60h, C = 1.

NEG: Az operandusának ellentettjét számolja ki.

AAM: nincs operandusa. Az AL értékét állítja át binárisan kódolt decimális (BCD) értékre AH-ba és AL-be.

DIV: Előjel nélküli osztás. Egyetlen operandusa van, amely nem lehet közvetlen operandus (vagyis konstans). Ha az op operandus 8 bites, akkor az eredmény úgy áll elő, hogy AL tartalmazz az AX/op hányadosát, AH-ba pedig AX/op maradéka kerül. Ha az op operandus 16 bites, akkor AX-be a (DX:AX)/op hányadosa, DX-be pedig a (DX:AX)/op maradéka kerül.

IDIV: Előjeles osztás. Egyetlen operandusa van, amely nem lehet közvetlen operandus (vagyis konstans). A nem 0 maradék előjele megegyezik az osztóéval. Ha az op operandus 8 bites, akkor az eredmény úgy áll elő, hogy AL tartalmazz az AX/op hányadosát, AH-ba pedig AX/op maradéka kerül. Ha az op operandus 16 bites, akkor AX-be a (DX:AX)/op hányadosa, DX-be pedig a (DX:AX)/op maradéka kerül.

Az osztásnál előfordulhat, hogy a hányados nem fér el AH-ban vagy AX-ben, ilyenkor azonnal abortál a program. Célszerű megelőző ellenőrzéseket végezni:

- Nem nulla-e az osztó?

- Nem túl nagy-e az osztandó? (Előjel nélküli osztásnál túl nagy az osztandó, ha AH > op).

CMP: összehasonlító művelet, két operandusa van op1 és op2. Az op1 - op2 szerint állítja be a flag-eket.

Adatmozgató utasítások

Tekintettel arra, hogy néhány utasítás alapvetően az AL-ben, az AX-ben vagy a DX:AX-ben szereplő értékeket veszi első operandusának, nem csak értékadásra, hanem regiszterek közötti adatok cseréjére is szükség lehet. Bizonyos esetekben nincs olyan felesleges üres általános regiszter ahová át lehet tölteni az adatot, ilyenkor a vermet használjuk. Az adatmozgató utasítások a POPF és SAHF kivételével nem módosítják a flag-eket.

XCHG: két operandusa van, amelyek nem lehetnek közvetlen operandusok. Az utasítás kicseréli a két operandusának tartalmát egymással.

XLAT: nincs operandusa. Az AL-be a [BX+AL] által címzett maximum 256 byte-os tartomány AL-edik bájtját tölti.

PUSH: egyetlen operandusát betölti a veremmutató (SS:SP) által mutatott memória címre a verembe, majd a veremmutatót két bájttal csökkenti (a verem „lefelé bővül").

PUSHF: nincs operandusa, a STATUS regiszter értékét tárolja el a verembe (Push Flags).

POP: egyetlen operandusa a veremmutató (SS:SP) által címzett értéket másolja az operandusként megadott helyre.

POPF: nincs operandusa, a STATUS regiszter értéke lesz a veremmutató által mutatott memóriacím tartalma (Pop Flags). A veremmutató ezután 2-vel növekszik.

SAHF: Az AH regiszter tartalma átírodik a STATUS regiszter alsó 8 bitjébe.

LAHF: Az STATUS regiszter alsó 8 bitje átírodik az AH regiszterbe.