Assembly programozás levelező tagozat

Németh Gábor

Szegedi Tudományegyetem
Képfeldolgozás és Számítógépes Grafika Tanszék

2011-2012-2

Tematika

- Assembly nyelvi szint. Az Intel 8086/88 regiszter készlete, társzervezése, címzési módjai, címzési mód byte.
- Az Intel 8086/88 utasításai (adat mozgató, aritmetikai, logikai, string kezelő, bit léptető/forgató, vezérlés átadó, processzor vezérlő, input/output utasítások, program megszakítás, szemafor).
- Pszeudo operátorok. Egyszerű adat definíciós utasítások. Struktúra, rekord (definíció, hívás, mezőre hivatkozás).
- Eljárás (deklaráció, hívás, paraméter átadás/átvétel). Lokális adat terület, rekurzív és reentrent eljárások.
- Feltételes fordítás.
- Makró (definíció, hívás), blokk ismétlés.
- Címkék, változók, konstansok, kifejezések.

Tematika

- Szegmens definíció, szegmens csoport, aktív szegmensek kijelölése, globális szimbólumok.
- Assemblernek szóló utasítások, lista vezérlési operátorok.
- Egyszerűsített lexikális elemző.
- Két menetes assembler fordító.
- Makró generátor.
- Szerkesztő.
- Time sharing (idő osztás). Binding (cím hozzárendelés), dinamikus szerkesztés.
- Programok hangolása.

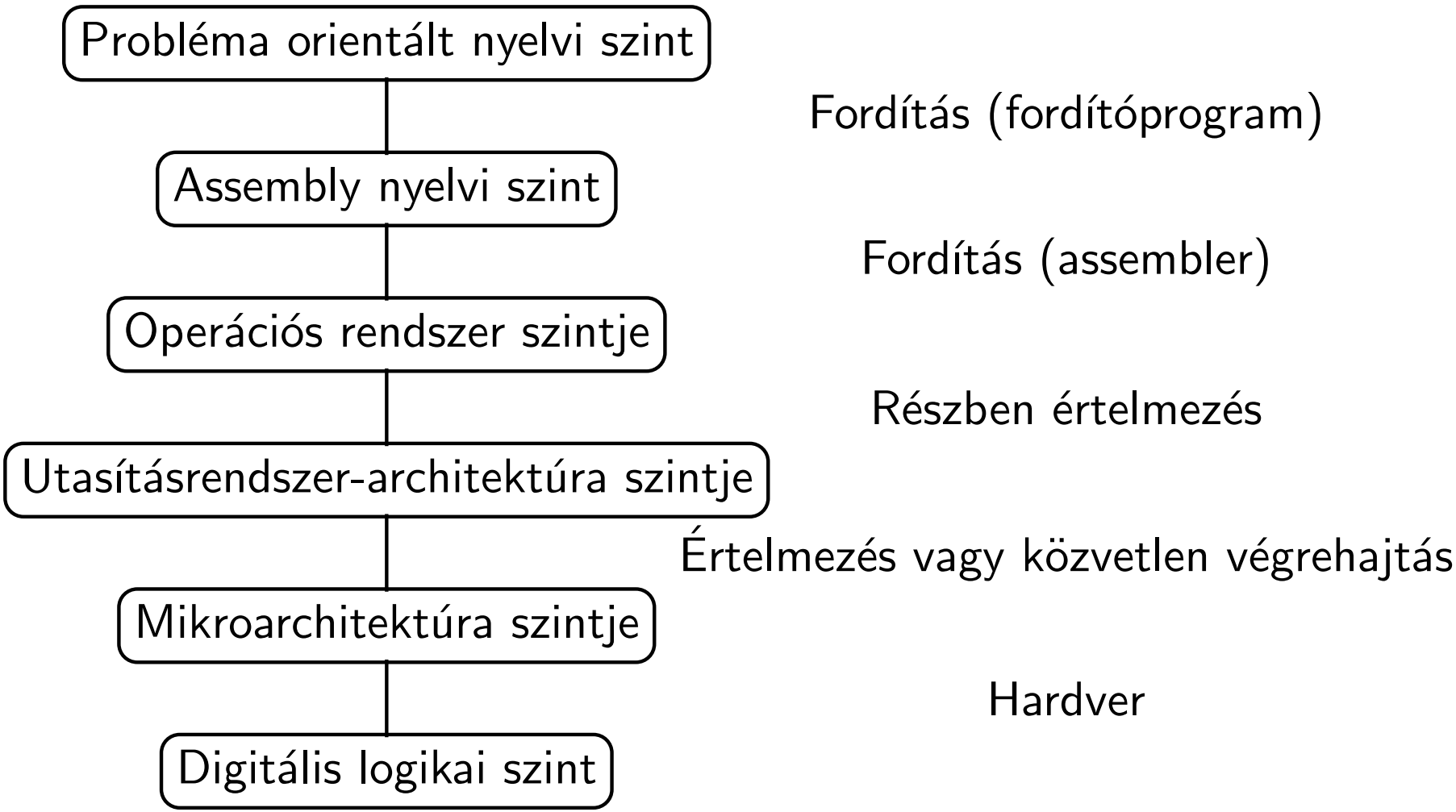
Ajánlott irodalom

- Máté Eörs: Assembly programozás (NOVADAT, 1999, 2000).
- Pethő Ádám: IBM PC/XT felhasználóknak és programozóknak, 1. kötet: Assembly alapismeretek (SZÁMALK, 1992).
- S. Tanenbaum: Structured computer organization (Prentice Hall, 2006). Magyarul: Számítógép-architektúrák 2. átdolgozott, bővített kiadás (Panem 2006).
- B. B. Brey: Programming the 80286, 80386, 80468, and Pentium-baseb Personal Computer (Prentice Hall, 1996).

Követelmények

- Gyakorlat:
  - 1 nagy ZH: 90 perc, 40 pont
  - a gyakorlat sikeres, ha a hallgató legalább 20 pontot szerzett
  - beadható feladat
- Kollokvium:
  - írásbeli vizsga
  - kollokviumjegy: (lásd a CooSpace-ben és az ETR-ben)

Architektúrális szintek



Fordítás és értelmezés

Fordítás: magasabb szintű nyelvek átalakítása numerikus gépi nyelvre vagy annak szimbólikus formájára.
Értelmezés: A utasítás sorozatok beolvasása és végrehajtása.
Assembler: olyan fordító ahol a forrásnyelv szimbólikus formája a gépi nyelvnek.

Magas szintű nyelvek

Változók, függvények, típuskonverziók, polimorgizmus, objektum-orientáltság

```
double sum = 0;
for ( int i = 0; i < 20; i++ ) {
sum += sqrt( (double) i );
}
```

Assembly nyelv

Egysoros utasítások. Az utasítások a gépi kód szimbólikus kódja

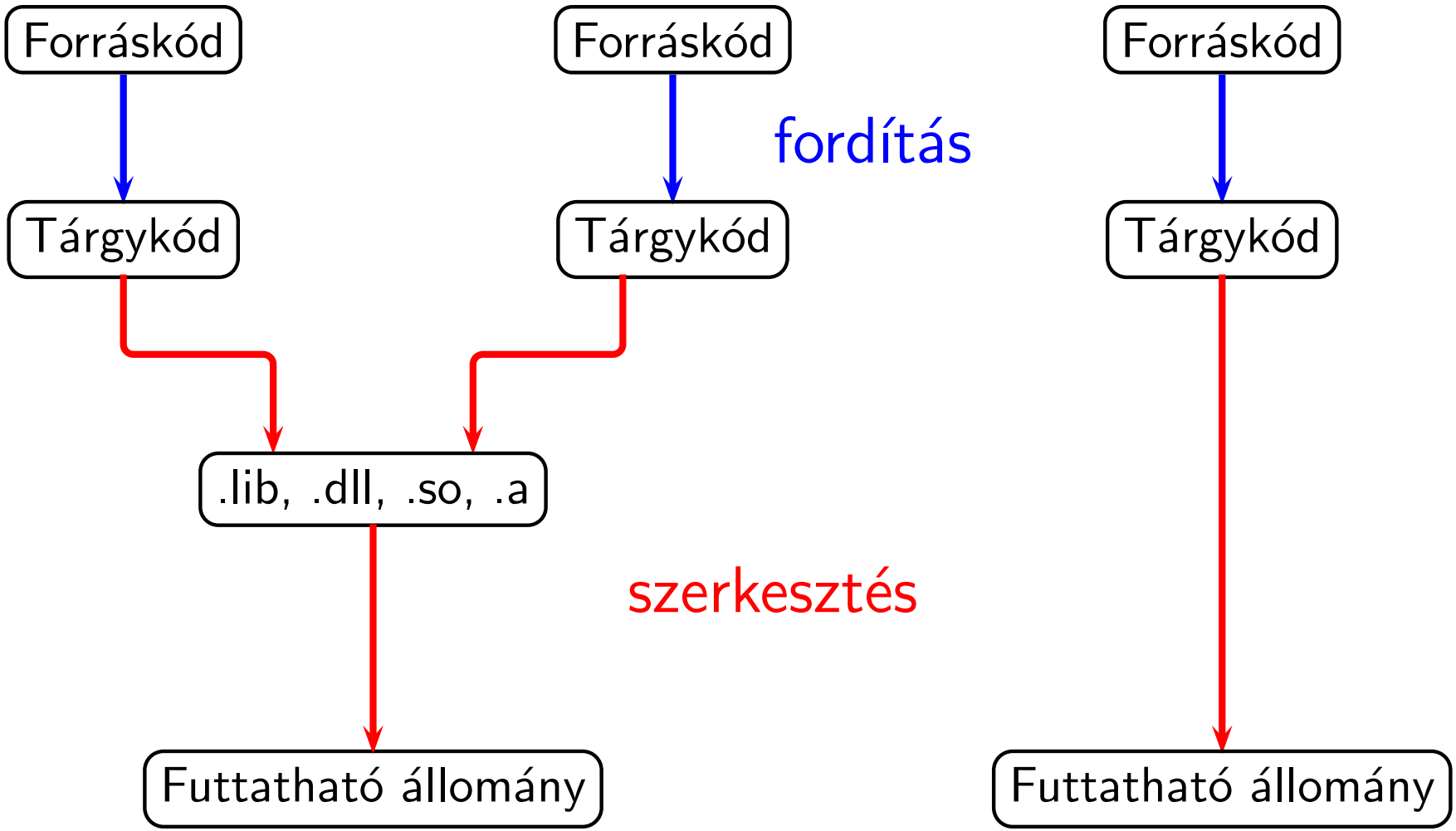
```
MOV AX, 10d ; AX = 10d
ADD AX, [02] ; AX=AX+DS:[02]
PUSH AX ; AX a verembe kerul
```

Gépi nyelv

- A gép számára érthető.
- Gyorsan végrehajtható.
- Processzortípusonként változhat.

```
00 05 0F 2B 42 6F
24 23 94 6E 4A B2
```

Forráskódból futtatható állomány előállítása



Néhány jellemző

- Little endian memória-szervezés
- 16 bites regiszterek
- 20 bites címbusz

Regiszterek

Szegmens regiszterek (16 bites regiszterek)

- CS (Code Segment): utasítások címzéséhez
- SS (Stack Segment): verem címzéséhez
- DS (Data Segment): adat terület címzéshez
- ES (Extra Segment): másodlagos adatterület címzéshez

Regiszterek

Vezérlő regiszterek (16 bites regiszterek)

- IP (Instruction Pointer): az éppen végrehajtandó utasítás logikai címét tartalmazza a CS által mutatott szegmensben
- SP (Stack Pointer): a verem tetejére beírt adat logikai címére mutat az SS által mutatott szegmensben
- STATUS (vagy SR, vagy FLAGS): a processzor állapotát jelző regiszter
- BP (Base Pointer): a verem indexelt címzéséhez használatos
- SI (Source Index): a kiindulási adat terület indexelt címzéséhez használatos
- DI (Destination Index): a cél adat terület indexelt címzéséhez használatos

Regiszterek

Általános regiszterek 16-bites regiszterek.
Az általános regiszterek felső 8 bitje és alsó 8 bitje külön is címezhető

regiszter	felső bájt	alsó bájt	
AX	AH	AL	Accumulator (szorzás, osztás)
BX	BH	BL	Base Register (címző regiszter)
CX	CH	CL	Counter Regiszter (számláló)
DX	DH	DL	Data Register (szorzás, osztás, I/O)

Címzési módok

- Szegmensekre (egybefüggő területekre) osztott memóra.
- A szegmens címek mindig oszhatóak 16-tal (paragrafushatár).
- Virtuális cím: `BÁZISCÍM:OFFSZET`
- Fizikai cím: a fizikai címet a szegmens kezdőcíme és a az offszet együttesen határozza meg.
  `BÁZISCÍM` $\cdot$ `16 + OFFSZET` alsó 20 bitje. (valós módban)
- Egy fizikai címhez több logikai cím tartozik.

Adat terület címzés

Kódba épített adat Az adatot közvetlenül a regiszterbe írjuk. Ennél a címzési módnál csak konstansokat tudunk megadni második operandusként.

Formátum: `regiszter, konstans`
Például:

- `AX, 6`
- `AX, 06F2h`

Adat terület címzés

Direkt memória címzés A címrészen az operandus logikai (offszet) címét adjuk meg, nem az adatot.
Formátum: `regiszter, memóriacím`
Például:

- `AX, SZO` , *ahol a SZO egy word értéket tartalmaz*
- `AL, KAR` , *ahol a KAR egy byte értéket tartalmaz*

Ügyelni kell arra, hogy a két operandus mérete összhangban legyen egymással. 16 bites regiszterhez csak 16 bites (word) címeket használhatunk, 8 bites regiszterhez csak 8 bites címeket használhatunk!
Direkt címzésnél megadhatunk második operandusként egy regisztert is. Akár egy regisztert, akár egy logikai címet adunk meg, mindkét esetben a memóriacímen tárolt adat változhat, viszont a cím azonos marad.

Adat terület címzés

Indexelt címzés Az operandusban megadott 8 vagy 16 bites számot (eltolás) hozzáadjuk az index-regiszter (SI vagy DI) tartalmához, így alakul ki a logikai cím.
Formátum: `regiszter, szám[index-regiszter]`
Például:

- `AX, [SI]` , *itt csak az SI-ben tárolt értéket számítjuk*
- `AX, 10h[SI]` , *SI tartalmához hozzáadjuk a 10h konstanst*
- `AX, -10h[SI]` , *SI tartalmához hozzáadjuk a -10h konstanst*

A 8 biten megadott eltolás érték előjel helyesen 16 bitre bővül a cím kiszámításakor.

Adat terület címzés

Regiszter indirekt címzés Az operandusban a címet egy regiszter tartalmazza. Az ilyen módon megadott címet mutatónak hívjuk. Nem adható meg eltolás. Ebben a címzési módnál csak az SI, DI és BX regiszter használható.
Formátum: `regiszter, [regiszter]`
Például:

- `AX, [SI]` , *az SI-ben tárolt értéket használjuk*
- `AX, [BX]` , *a BX-ben tárolt értéket használjuk*
- `AX, [DI]` , *a DI-ben tárolt értéket használjuk*

Adat terület címzés

Bázis relatív címzés Az operandusban megadott logikai címet úgy kapjuk meg, hogy az eltolás értékét, az SI vagy DI valamelyikének értékét, valamint a BX regiszterben tárolt értéket összeadjuk.
Formátum: `regiszter, szám[regiszter][regiszter]`
Például:

- `AX, 10h[SI][BX]`
- `AX, [BX][DI]`
- `AX, [BX + SI + 10h]`

Verem terület címzés

A verem (stack) terület címzése bázis relatív címzéssel történik, azzal a különbséggel, hogy BX helyett BP-t használjuk. A fizikai cím meghatározásához nem DS-t, hanem SS lesz a szegmens regiszter.

Program terület címzés

Egy-egy utasítás során az IP értéke az utasítás hosszával növekszik és a soron következő utasításra mutat a CS-ben. Bizonyos vezérlési szerkezetekben (pl. ciklusok, feltételek) az IP értékének megadásával a kívánt helyen folytathatjuk a program futását.

Program terület címzés

IP relatív címzés Az IP (már módosult) pillanatnyi értékéhez hozzáadódik a 8 bites előjeles operandus. A programkódokban bizonyos kódrészletek kezdő utasítását címkével láthatjuk el, ha azt szeretnénk, hogy egy adott feltétel esetén ott folytatódjon a program futása. A feltételes vezérlés átadás és a ciklus utasítás mindig ilyen címzési móddal valósul meg.
Például:
`JMP CIKLUS` , *ahol CIKLUS egy címke az assembly forráskódban*

Program terület címzés

Direkt utasítás címzés Ez a címzési mód közeli (NEAR) vagy távoli (FAR) vezérlés átadásoknál játszik szerepet. A vezérlés átadás közeli, ha a programkód ugyanabban a szegmensben folytatódik tovább és távoli szegmensváltás esetén. Közeli vezérlésátadás esetén a 16 bites operandus lesz az IP új tartalma, távoli vezérlésátadásnál pedig az IP és a CS tartalma is megváltozik. Ilyen vezérlés lehet például az eljáráshívás.
Például:
`CALL ELJARAS` , *az ELJARAS nevű eljárás hívása*

Program terület címzés

Indirekt utasítás címzés Bármilyen címzési móddal megadott szóban vagy dupla szóban tárolt címre történő vezérlés átadás. Például:

- `JMP AX`
- `JMP [BX]`

Az assembly utasítások szerkezete

Assembly programozási nyelvben egy utasítás a következő sémát követi:

```
címrész operációs_kód operandusok kommentár
```

- címrész: egyes adatok illetve utasítások szimbólikus jelölése
- operációs_kód: mnemonic, az utasítás, művelet megnevezésére szolgál
- operandusok: az utasítás paraméterei
- kommentár: A program jobb olvashatóságát és érthetőségét teszi lehetővé, de nincs hatása a program működésére

Példa:

```
hat:   MOV   AX, 6   ; ide ugrik a vezerles
```

címrész — utasítás_kód — operandusok — kommentár

Az assembly utasítások szerkezete

prefixum	operációs kód	címzési mód	operandus
0-2 byte	1 byte	0-1 byte	0-4 byte

- Prefixum: utasítás istmétlés (sztringkezelő utasítások), explicit szegmens megadás (pl. `CS:S`) vagy `LOCK` (szemafor).
- operációs kód: a művelet kódja (a szimbólikus alakja a mnemonic)
- címzési mód byte: hogyan kell az operandust címezni
- operandus: mivel kell az operandust elvégezni

Címzési mód byte

7	6	5	4	3	2	1	0
Mód		Regiszter			Reg/Mem		

Adatmozgató utasítások

- `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.
- MOV: két operandusa van op1 és op2. Az op2 értékét op1-be viszi

Adatmozgató utasítások

- `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.

A STATUS regiszter

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	O	D	I	T	S	Z	-	A	-	P	-	C

Overflow	Előjeles túlcsordulás
Direction	a string műveletek iránya, 0: növekvő, 1: csökkenő
Interrupt	1: megszakítás engedélyezése, 0: tiltása
Trap	1: „single step”, 0: automatikus üzemmód
Sign	az eredmény legmagasabb helyiértékű bitje (előjel)
Zero	1, ha az eredmény 0, különben 0
Auxiliary Carry	átvitel a 3. és 4. bit között (decimális aritmetika)
Parity	az eredmény alsó 8 bitjének paritása
Carry	átvitel előjel nélküli műveletekhez

Összeadás

- `ADD op1, op2`
- `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.
- `AAA`: A pakolatlan (ASCII kódú, egy számjegy nyolc biten) decimális számok összeadását támogatja.
- `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`.

Kivonás

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