Számítógép architektúrák alapjai

Hallgatói jegyzet

Számítási modell és architektúra

2023

Tartalom

1. Számítási modell 1
1.1. Bevezetés 1
1.2. Fogalma 1
1.3. A számítási modell, programnyev és architektúra kapcsolata 1
1.4. Csoportosítás 1
1.5. Adatalapú modellek 2
1.5.1. Neumann modell 2
1.5.2. Adatfolyam modell 3
1.5.3. Összehasonlítás 4
1.6. Applikatív modell 5
2. Architektúra 6
2.1. Bevezetés 6
2.2. Logikai architektúra 6
2.3. Fizikai architektúra 6

Készítette: Kováts Máté

A jegyzet Durczy Levente 2019 őszi Számítógép architektúrák alapjai előadás videói alapján készült.

Segítséget jelentett Uhrin Ádám és Nagy Enikő korábbi jegyzetei.

1. Számítási modell

1.1. Bevezetés

A processzor architektúrák az 1970-es években kezdtek el robbanásszerűen fejlődni (pl.: Intel 8080), amelynek eredményeképpen olyan innovatív nyelvek és architektúrák jelentek meg int például RISC, CISC, szuperskalár architektúrák (lásd: Korszerű architektúrák I.), melyek már eltértek az 1950-es években kidolgozott Neumann-elvektől, azon belül is a legalapvetőbb részétől, a szekvenciális utasításvégrehajtástól. Viszont a koncepciójuk elsősorban a számítási modellben különbözik. Tehát ennek vizsgálatával jól megkülönböztethetőek az egyes architektúrák.

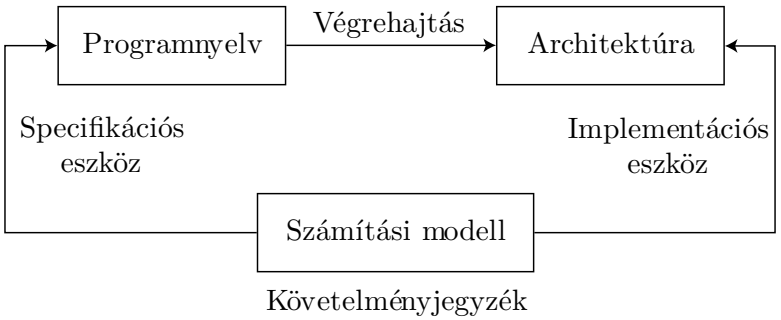
1.2. Fogalma

A számítási modell a számításra vonatkozó alapelvek egy absztrakciója.

Milyen absztrakciós jellemzők vannak, ami alapján megkülönböztetjük a számítási modelleket:

1. Min hajtjuk végre a számítást?
2. Hogyan képezzük le a számítási feladatot?
3. Milyen módon vezéreljük a végrehajtási sorrendet?

1.3. A számítási modell, programnyev és architektúra kapcsolata



Az ábrán látható három absztraktív részekből építjük fel a számítógépet. A számítási modellt úgy is nevezhetünk, hogy ez egy követelményjegyzék. Amikor megtervezünk egy rendszert először a számítási modellt tervezzük meg, ami megmondja mit akarunk csinálni. Ehhez kell egy specifikációs eszköz, ami tulajdonképpen a gyakorlati probléma megvalósítási eszköze, amikor is a követelmény jegyzék formalizálásra kerül. Ez maga a programnyelv. Ezt követi a végrehajtás, az architektúra. Az architektúra egy implementációs eszköz, ami biztosítja, hogy a követelményjegyzék és a hozzá készített programnyelv ezen az implementáción megfelelően működjön.

1.4. Csoportosítás

Alapvető számítási modellek csoportosítva:

1. Számítási modelljük szerint:
    a. szekvenciális
    b. párhuzamos

2. Vezérlés meghajtása szerint:
    a. vezérlés meghajtott
    b. adat meghajtott
    c. igény meghajtott
3. Probléma leírás szerint
    a. procedurális
    b. deklaratív

A három absztrakciós jellemzőből főként az alapján csoportosítjuk a modelleket, hogy min hajtjuk végre a számítást:

1. Adat alapú modellek
    a. Neumann modell
    b. Adatfolyam modell
    c. Applikatív modell
2. Objektum alapú modellek
3. Predikátum logika alapú modellek
4. Tudás alapú modellek
5. Hibrid modellek

A mai desktop és mobil architektúrákban olyan hibrid működik, ami keveri a Neumann és adatfolyam modell elemeit.

1.5. Adatalapú modellek

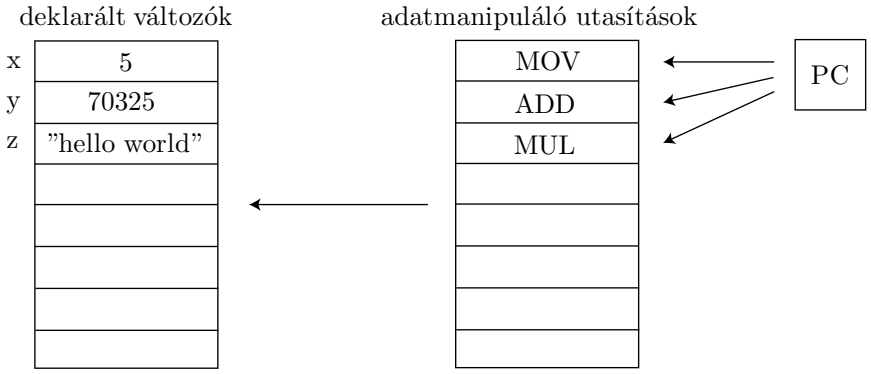
Legfontosabb közös tulajdonságuk, hogy az adatok általában típussal rendelkeznek, azon belül is elemi (pl.: integer 16 bit) vagy összetett adattípussal. Az adattípus meghatározza az értelmezési tartományt, értékkészletet és az adaton elvégezhető műveletek halmazát. Ezeket a követelményjegyzékben és a programnyelvben pontosan meg kell határozni.

1.5.1. Neumann modell

Absztrakciós jellemzők alapján:

1. Min hajtjuk végre a számítást:
    - A számítást adatokon hajtjuk végre
    - Az adatokat változók képviselik
    - Változók korlátlan számban változtathatnak értéket (többszörös értékadás)
    - Adatok és utasítások azonos memóriaterületen helyezkednek el
    - A számítási feladat elemi műveletek sorozataként értelmezhető
2. Hogyan képezzük le a számítási feladatot:

    Adatmanipuláló utasítások sorozatával: a deklarált változók és adatmanipuláló utasítások a memóriában helyezkednek el. Ezek az adatmanipuláló utasítások szekvenciálisan megváltoztatják a változók értékét. A folyamat egy dedikált regiszter (Program Counter - PC) segítségével történik, ami mindig tartalmazza a következő utasítás címét. Ez az implicit statikus szekvencia.



3. Milyen módon vezéreljük a végrehajtási sorrendet:

   Az implicit statikus szekvenciától eltérni csak explicit vezérlésátadó utasításokkal lehet, melyek arra szolgálnak, hogy megváltoztassák a PC értékét. Mivel a PC minden végrehajtás után inkrementál, ezáltal a következő utasítás címére fog mutatni, következésképpen vezérlésátadó utasításokkal befolyásolható a végrehajtási sorrend. Ezeknek tudatában hívjuk a Neumann modellt vezérlés meghajtottnak. A vezérlést a PC biztosítja, a végrehajtás sorrendjét pedig a programozó.

Következményei:

- előzmény érzékenység: mivel a változók korlátlan számban változtathatják az értéküket, így nem mindegy, hogy az adott változót melyik utasítás után tekintjük, az aktuális állapota változhat.
- alapvetően szekvenciális végrehajtást biztosít
- egyszerűen implementálható adatmanipuláló utasítások állapot módosulást okozhatnak: bizonyos utasítások nem szándékolt állapot módosulást okozhatnak, például két 16 bites integer számot összeszorzunk túlcsordulás keletkezhet. Ezeket az úgynevezett mellékhatásokat kezelni kell a programvégrehajtás során.

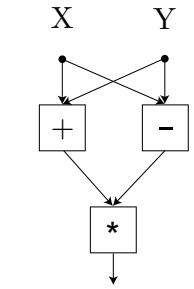
1.5.2. Adatfolyam modell

Absztrakciós jellemzők alapján:

1. Min hajtjuk végre a számítást: ebben az esetben is adatokon:
   - Az adatokat bemenő adathalmaz képviseli
   - Egyszeres értékadás lehetséges
2. Hogyan képezzük le a számítási feladatot:

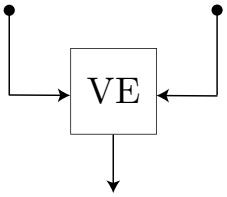
   Adatfolyam gráf és input adatok halmaza segítségével.

   Adatfolyam gráf jelentését a következő példával lehet szemléltetni. Van két deklarált változónk, X és Y, illetve Z=(X+Y)*(X-Y) függvény. A gráf élekből és csomópontokból épül fel. A csomópontok jelentik a műveleteket, az élek pedig az adatutakat, melyeken folynak az adatok. Bemenő adatok ez esetben X és Y. Az élek mentén van egy összeadás és kivonás művelet, majd a kimenetük bemenetként szolgál a szorzáshoz, melynek kimenetén előáll az eredmény.



Z = (X+Y)*(X-Y)

Fontos észrevenni, míg a Neumann modellnél az összes utasítást szekvenciálisan egy darab műveletvégző hajtotta végre. Adatfolyam modell esetében szakosodott végrehajtó egységek vannak, melyek kicsik, egyszerűen megvalósíthatóak és gyorsak. Szakosodott végrehajtó egyégek következménye, hogy csökken a végrehajtási idő, mivel a műveletek párhuzamosan is végezhetőek. Példánkban Z előállítása Neumann modell esetén 3 lépésben (ADD, SUB, MUL), adatfolyam modell esetén 2 lépésben (ADD és SUB párhuzamosan, MUL) elvégezhető, mely ilyen kis feladat esetén is 33%-os végrehajtási idő csökkenést jelent.



3. Milyen módon vezéreljük a végrehajtási sorrendet:

   Adatvezérelt (Stréber modell), vagyis a végrehajtási sorrendet az adatok elérhetősége határozza meg. A végrehajtó egységek várják a bemenő adatokat és minden végrehajtó egység csak akkor fog működésbe lépni, ha bemenetén minden bemenő operandus megjelenik. Amint megjelenik, azonnal végrehajtja a műveletet, nem vár utasításra, nem vár PC-re (mivel nincs). Az adatvezérelt program utasításai semmilyen szempontból nem rendezettek és az adatok utasításon belül vannak tárolva.

1.5.3. Összehasonlítás

	Neumann modell	Adatfolyam modell
Min hajtjuk végre a számítást	adatokon	
Adatokat mik képviselik	változók	bemenő adathalmaz
Értékadás	többszörös	egyszeres
Adattárolás helye	közös operatív tár	utasításon belül (regiszterek)
Számítási feladat leképezése	adatmanipulációs utasítások	adatfolyam gráf
Végrehajtás	szekvenciális	párhuzamos (sok műveletvégző, azonnali műveletvégzés, szakosodott végrehajtó egységek)

Végrehajtás vezérlése	vezérlés meghajtott: implicit szekvencia (PC), explicit vezérlésátadás	adatvezérelt: azonnali műveletvégzés amint lehetséges, adatok és utasítások nem rendezettek
Végrehajtás jellege	procedurális	
Következmények	előzményérzékeny	nincs előzményérzékenység
Implementáció	egyszerű	nehézkesebb (magasabb kommunikációs és szinkronizációs igény)

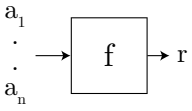
1.6. Applikatív modell

Fő jellegzetessége, hogy a számítási feladat egy komplex függvény formájában van megadva, ami argumentumok adathalmazára vonatkozik.

Például 'r' eredmény az 'f' komplex függvény 'a_1, …, a_n' input argumentumokkal írható le, ahol f általában egy matematikai függvény.

$$r = f(a_1, \ldots, a_n)$$

Végrehajtóegységgel ábrázolva:



A számítás adatalapú, viszont deklaratív jellegű, ami annyit jelent, hogy valamennyi a probléma megoldásához szükséges tényt és relációt előre deklarálunk. ezt a tény és reláció halmazt egy végrehajtó mechanizmus feldolgozza. A vezérlés igény meghajtott (Lusta modell), vagyis a végrehajtóegységek csak akkor fognak működésbe lépni, ha a számítási folyamat az eredményre igényt tart. Applikatív modell egyik jellemző architektúrája a redukciós architektúra.

2. Architektúra

2.1. Bevezetés

Mit is jelent az architektúra fogalma? 1964-ben Amdahl, az IBM mérnöke használta ezt a kifejezést. Ami azt mondta, hogy az architektúra az az ismeret halmaz a processzor belső felépítéséről, amit a gépi kódú programozónak ismernie kell, hogy hatékony programot tudjon írni. Az 1970-es években Bell és Newell négy szintet azonosítottak az architektúra fogalmának a meghatározásában:

- PMS (processzor, memória, switchek), ami magát a számítógépet és környezetét írja le
- programozási szint: magas (Python, JavaScript) és alacsony (C, Assembly) szintű programozás
- logikai tervezési szint
- áramköri szint

Másik megfogalmazásban a külső jellemzők, belső felépítés és működés együttesét jelenti. Az architektúra fogalmánál egy adott L absztrakciós szinten megnézzük az architektúrának M számítási modelljét, vesszük az S specifikációt és I implementáció összegségét $\{M, S, I\}_L$

További felosztásban logikai és fizikai architektúráról beszélhetünk.

2.2. Logikai architektúra

Logikai architektúra az egy meghatározott leírási szinten a programozó által látott specifikáció $\{M, S\}_L$ az úgynevezett funkcionális leírás (fekete dobozként tekintünk az architektúrára).

A számítógép szintű logikai architektúra rendszerszintű, azt vizsgáljuk milyen bemenetekre milyen eredményt produkál. Központi eleme az operációs rendszer, melyen keresztül vizsgálható.

Processzor szintű architektúra esetén ugyanúgy egy fekete dobozként tekintünk a processzorra, és a programozó feladata, hogy olyan bementet adjon amire az elvárt kimenet kapható. A processzornak van egy utasításkészlete (ISA). Minden programnyelven írt utasítás lefordítódik a processzor utasításkészletére és az alapján hajtódik végre. Komponensei:

- adattér
- adatmanipulációs fa
- állapottér
- állapotműveletek

2.3. Fizikai architektúra

A fizikai architektúra a modell és az implementáció együttes leírása adott absztrakciós szinten $\{M, I\}_L$

A számítógép szintű fizikai architektúra elemei:

- processzor
- memória
- buszrendszer

A processzor szintű fizikai architektúra elemei:

- műveletvégző egység
- vezérlő
- I/O rendszer
- megszakítás rendszer