Számítógép architektúrák alapjai

Hallgatói jegyzet

Megszakítási rendszer

2023

Tartalom

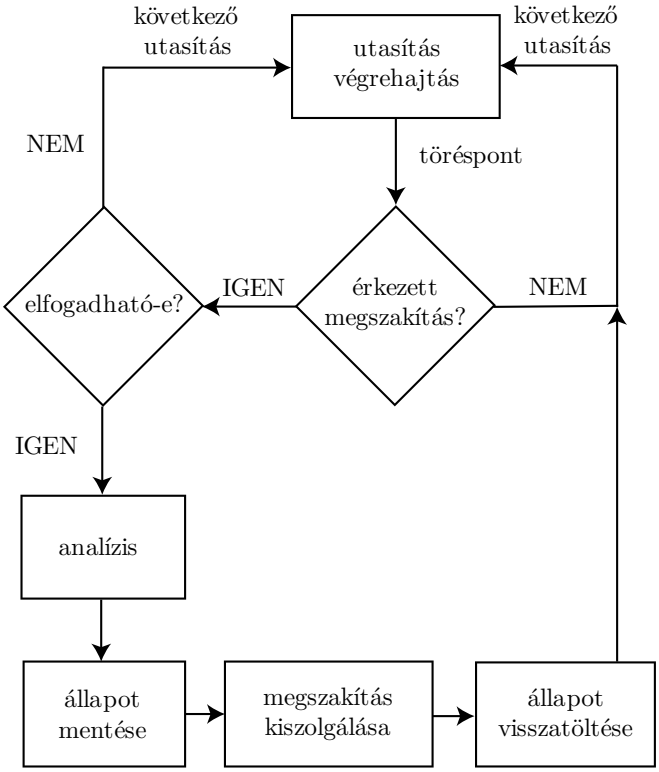
Készítette: Kováts Máté

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

Segítséget jelentett Uhrin Ádám és Nagy Enikő korábbi jegyzetei.

1. Megszakítási rendszer

A végrehajtás közben váratlannak tekinthető események kezelésére szolgál. Célja nem csak a reagálás, hanem a folyamatosan változó körülmények között az optimális működés biztosítása.



Ha CPU végzet az utasítás végrehajtással az utasítástöréspontban a processzor megvizsgálja, hogy érkezett-e megszakítás. Ha nem, akkor lehívja a következő utasítást. Ha érkezett megvizsgálja elfogadható-e. Amennyiben nem, szintén folytatódik a végrehajtás a következő utasítással. Viszont, ha elfogadható a megszakítás, akkor történik egy analízis, megtörténik az adott állapot (kontextus) mentése, hogy megszakítás kiszolgálása után visszatöltve folytatni lehessen a programot.

1.1. Történeti áttekintés

CPU és I/O-ból érkező adatok összehangolása céljából jött létre.

Kezdetben az I/O adatátvitel vezérlésére alkalmazták először, mivel a CPU rengeteg felesleges munkát végez azzal, hogy várakozik az I/O eszközökre a válaszra, mivel azok lényegesen lassabbak.

Később rájöttek, hogy a megszakítások belső rendszerek vezérlésére is alkalmasak. Megszakítás esetén a kontextus el kell menteni, majd betölteni a megszakítási rutin kontextusát, végre kell hajtani az megszakítást, majd visszaadni a vezérlést. Ez a kontextus tárolása gyakorlatban annyit tesz, hogy például a PC-be nem a következő utasítás címe, hanem a megszakítási rutin kezdőcíme kerül, majd PC eredeti állapotának visszatöltődik.

1.2. Okok és források

Prioritási sorrendben:

1. Géphibák: ezek a megszakítások nem letilthatóak
   Például: automatikus hibafigyelő áramkörök jelzései (hőszenzorok, energiaellátás, paritás, watch dog). Egy részük hibajelző kódok segítségével tárja fel a hibákat.
2. I/O források: ide tartozik minden perifériával kapcsolatos állapotjelzés
3. Külső források
   Például: reset gomb, hálózati kommunikáció
4. Programozási források:
   a. Szándékos: amikor egy program megszakítást kér. Például: rendszerhívás, BIOS hívás
   b. Nem várt (hibakezelés): mindig valamilyen utasítás végrehajtása vagy a végrehajtás megkísérlése során alakul ki

1.3. Szoftveres megszakítások

A következő hibáknál léphet fel:

1. Memóriavédelem megsértése
   Alapértelmezetten minden program mikor fut, lefoglalódik számára egy memória terület. Amennyiben a program egy utasítása ezen a memóriaterületen túlmutat az OS megszakítást kezdeményez.

2. Tényleges tárkapacitás túlcímzése
   Adott implementációnál amennyiben az adott program egy olyan címet próbál elérni, amely fizikailag nem létezik, megszakítás következik be.

3. Címzési előírások megsértése
   Példa: a címek 32 bitesek és a címzés byte-osan történik. Ebben az esetben csak minden negyedik byte címezhető. Ha program a címzésnél ezt megsérti, megszakítás történik.

4. Aritmetikai logikai végrehajtás közben fellépő hibák
   Például: overflow, underflow, 0-val való osztás

1.4. Csoportosítása

1. Megkülönböztetünk szinkron és aszinkron megszakításokat.
   a) Szinkron megszakítás: adott programnak ugyanazon paraméterével történő futtatása során mindig ugyanott jelentkezik a megszakítás (várható megszakítás). Például: DMA kérés
   b) Aszinkron megszakítás: véletlenül lépnek fel. Például: hardverhibák

2. Utasítások végrehajtása között vagy utasítások végrehajtása közben lépett fel.
   a) Között: egy utasítás végrehajtásának eredményeképpen következik be. Utasítás végrehajtás után azonnal elkezdődhet a kezelés. A program folytatása a kezelés eredményétől függ. Például: overflow, tárvédelem
   b) Közben: egy utasítás végrehajtása alatt következik be. Nincs szinkronban a végrehajtási ciklussal. Például: hardvermegszakítások

3. Felhasználó által kért és nem kért megszakítások
    a) Kért megszakítás például: OS rutinok, BIOS rutinok
    b) Nem kért például: overflow, I/O befejezés, debug

4. A megszakított program a megszakítás után folytatódik vagy befejeződik
    a) Folytatódik például: I/O megszakítás, OS rutinok
    b) Nem folytatódik például: hardverhiba

5. Maszkolható vagy nem maszkolható megszakítások
    a) Maszkolható: a megszakítást le lehet tiltani. Prioritás rendelhető hozzá és bizonyos esetekben a megszakítás nem lép érvénybe. Például: I/O kérés
    b) Nem maszkolható: hardverhiba (túlmelegedés), paritásbit jelzések

1.5. Megszakítás kiszolgálás általános folyamata

Egy megszakítás segítségével egy adott program leállítható és tetszőleges program lefuttatása után folytatható.

Kiszolgálás előkészítése:

- Ha valamelyik egység megszakítás kérést bocsát ki, az aktiválja az INTR vezérlő vonalat
- Első kérdés: elfogadható-e a megszakítás?
- A processzornak van egy magszakítás bemenetük, ezen érzékelik a megszakítást
    - Régebben ez egyszintű: van vagy nincs megszakítás
    - Jelenleg megszakítás áramkör: tudja kezelni a prioritásokat → ez van rákötve a CPU INTR bemenetére
- Mikor juthat érvényre egy megszakítás? 3 feltétele van:
    - Megszakítható az aktuális folyamat (program vagy megszakítás)
    - Megfelelő a prioritás nagysága
    - Az adott megszakítás nincsen maszkolva
- INTACK vezérlő vonal aktiválása, ha a megszakítás el van fogadva
- INTR vonal deaktiválása

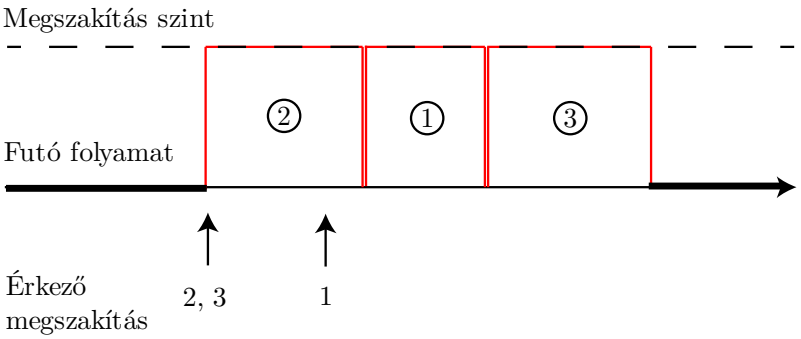
CPU feladata a megszakítás kiszolgálásakor eltárolni az adott folyamat kontextusát programtól független tárolóban. Betölti a PC-be a megszakítási rutin kezdőcímét és ha szükséges az állapotinformációkat is.

Különbség a közönséges program és megszakítás között, hogy a megszakított programtól teljesen függetlenül fut le egy folyamat.

1.6. Megszakítási rendszerek szintjei

1. **Egyszintű**: A megszakítás nem megszakítható, vagyis, ha egy megszakítás létrejön addig nem kezdődhet másik megszakítás, amíg vissza nem térünk a normál állapotba.

    Példa: legyen egy három szintű megszakítási rendszerünk. Prioritási szintek: $1 > 2 > 3$.



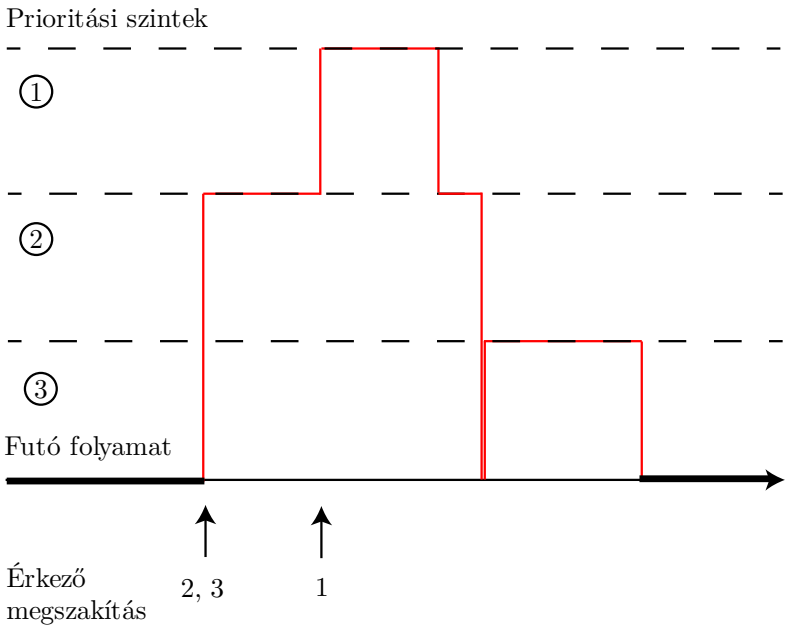
Egyidőben érkezik egy 2-3-as megszakítás kérés. Ebben az esetben a megszakítási rendszer a 2-est fogja előnyben részesíteni és az adott folyamatot a következő megszakítás töréspontan megszakítja, majd elkezdi végrehajtani. Mikor visszatér, azonnal elkezdi a 3-as megszakításkérés kiszolgálását. Amennyiben 2-es közben érkezik egy 1-es prioritású megszakításkérés, az aktuális megszakítást nem lehet megszakítani, viszont befejeztével az 1-es végrehajtása kezdődik meg, majd a 3-asé.

Függőleges vonalak: állapotmentések és betöltések.

Legnagyobb probléma az 1-es érkezésének időpontja, mert a legnagyobb prioritású megszakítás kérés várakozni kényszerül.

2. **Többszintű**: A megszakítások is megszakíthatóak, ezért fontossági sorrendben futnak le.

Először a 2-es szintű megszakítás kezdi végrehajtani. Az érkező 1-es prioritás miatt a 2-est megszakítja és elkezdi végrehajtani a legmagasabb prioritásút. Befejeztével visszatölti a megkezdett 2-es megszakítást, majd a 3-ast.



Hátránya: a valóságban nagyon sok megszakítás van, nem lehetséges mindegyikhez külön prioritási szintet rendelni. Ezért kitalálták a többszintű, többvonalú megszakítási rendszert.

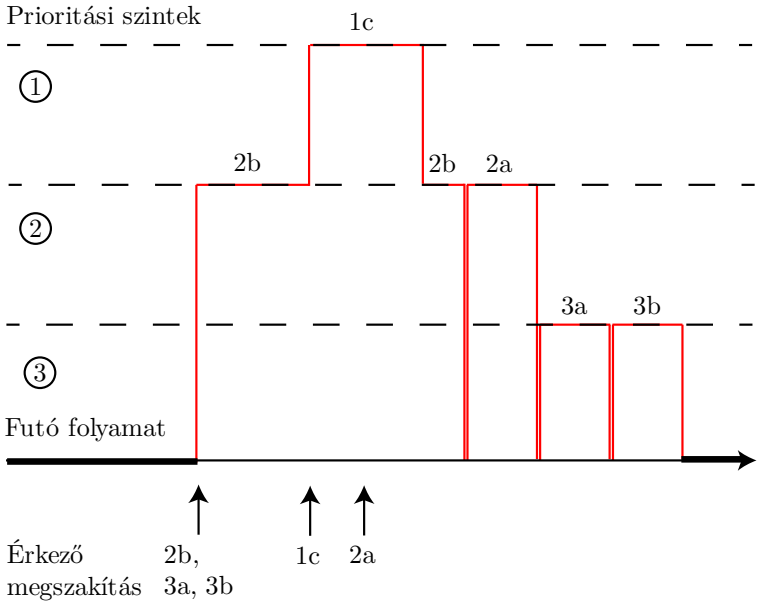
3. **Többszintű, többvonalú**: Elve, hogy a megszakítási forrásokat/okokat osztályozzák. A megszakítási osztályokhoz prioritási szinteket rendelnek.

Osztályok: 1, 2, 3

Prioritás: 1, 2, 3

Osztályokon belül megszakítás típusok: a, b, c, d ...

Osztályok között többszintű, osztályon belül egyszintű megszakítás rendszer. Egyszintű esetén is van prioritás.



Érkezik 2b, 3b, 3a megszakítás. 2b-t fogja végrehajtani a töréspontban. Eközben megjön 1c. Mivel magasabb prioritású, ezért megszakít és végrehajtja. Ha végzett, folytatja 2b végrehajtását. Közben érkezik egy 2a megszakításkérés és meg kell várnia míg 2b befejezi. Amikor 2a befejezte jöhet a 3a,majd a 3b (mert szinten belül számít a prioritás).

1.7. Összefoglalás

A megszakítás lényege, hogy csak akkor foglalja le a CPU figyelmét amikor arra tényleg szükség van.