Függelék

A következőkben találhatók azok az információk, melyek terjedelmi okok miatt nem kerültek be a dolgozat függelékébe, azonban az alkalmazás használatához és továbbfejlesztéséhez szükségesek lehetnek.

A feladatokhoz kapcsolódó algoritmusok

Az alábbiakban témakörönként összefoglalásra kerülnek a megvizsgált algoritmusok.

A taszkütemezési feladat algoritmusai

FCFS **FIFO**	**Legrégebben várakozó** **(First Come First Served / First Input First Output)**
	Mindig a legrégebben várakozó folyamatot választja ki futásra.
RR	**Körbeforgó (Round Robin)**
	Adott nagyságú időszeletet (time slice, quantum) biztosít a folyamatok számára. Az időszelet letelte után az ütemező elveszi a futás jogát és más folyamatot tesz futathatóvá. Ha a folyamat az időszeletnél kevesebb időt használt, akkor a löket (burst) végén történik az újraütemezés.
SJF	**Legrövidebb löketidejű (Shortest Job First)**
	A futásra kész folyamatok közül a legrövidebb becsült löketidővel rendelkező folyamatot választja ki futásra. Amennyiben egy taszk fut, az ütemező nem szakítja meg a kiválasztott taszk futását.
SRTF	**Legrövidebb hátralevő idejű (Shortest Remaining Time First)**
	Kibővíti az SJF algoritmust, mégpedig azáltal, hogy ha egy új folyamat kerül Futásra kész állapotba, akkor az ütemező megvizsgálja, hogy melyik folyamat hátralevő löketideje a kisebb és annak adja a futás jogát.

A memóriafoglalás algoritmusai

FF	**Első megfelelő (First Fit)**
	A tár elejéről induló algoritmus, az első rendelkezésre álló elegendő mérettel bíró szabad területből foglal helyet a folyamat számára
NF	**Következő megfelelő (Next Fit)**
	Az utoljára lefoglalt tartománytól indul, az első rendelkezésre álló elegendő mérettel bíró szabad területből foglal helyet a folyamat számára
BF	**Legjobban megfelelő (Best Fit)**
	A rendelkezésre álló szabad területekből a legkisebbet foglalja le
WF	**Legrosszabban illeszkedő (Worst Fit)**
	A rendelkezésre álló szabad területekből a lehető legnagyobbat foglalja le

A lapcsere algoritmusok

Optimális algoritmus (Optimal)	
OPT	Előre néző algoritmus, mely a lapok következő használatának idején alapul. A legoptimálisabb megoldást adja, azt a lapot választja kivitelre, amelyikre a legtovább nem lesz szükség. Hátránya, hogy megvalósítása nagyon bonyolult, közel lehetetlen.
Legrégebbi lap (First Input First Output)	
FIFO	A behozatal idejét vizsgálja. Azt a lapot választja ki, amelyik a legrégebben van a tárban. Sajnos gyakran cseréli le azokat a lapokat is, amelyekre a folyamatok gyakran hivatkoznak.
Újabb esély (Second Chance)	
SC	A FIFO-t kiegészítve figyeli a hivatkozott bit állapotát, ezáltal a használat tényét igyekszik figyelni. Hivatkozás esetén az R bitet törli, a lapot a FIFO végéhez fűzi, ellenkező esetben a lapot kiválasztja kivitelre.
Legrégebben nem használt (Last Recently Used)	

LRU	Azt a lapot választja ki áldozatnak, amelyre a leghosszabb ideje nem történt hivatkozás a folyamatok által. Hátrafele néző, lokalitást kihasználó, a használat idejét figyelembe vevő algoritmus, ami sajnos drága és hardvertámogatással működik hatékonyan. Több megvalósítása is lehetséges (számlálóval, láncolt listával, mátrixszal), itt azonban a legszemléletesebb, a számlálóval rendelkező megoldás kerül bemutatásra.
Legkevésbé használt (Least Frequently Used / Not Frequently Used)	
LFU **NFU**	Az algoritmus egy számlálót tart karban, melyet az algoritmus futásakor a számlálóhoz R bit aktuális értékét adja hozzá, ezután az R bitet törli. Az a lap kerül kiválasztásra, amelyik a legkisebb értékkel rendelkezik. Az algoritmus hátrányai (jelenleg nem, de egykor gyakran használt lapok tárban maradása és a frissen betöltött, kis számlálójú lapok könnyű kivitele) öregítéssel és tárba fagyasztással küszöbölhetők ki.
Utóbbi időben nem használt (Not Recently Used)	
NRU	Az algoritmus R és M bitek értékei alapján a lapokat 4 csoportba sorolja, majd a legkisebb prioritással rendelkező csoportból véletlenszerűen választ. Az R bit értéke idővel elveszíti jelentőségét, ezért az operációs rendszer idővel törli azt.

A kidolgozás során a következő algoritmus nem kerül implementálásra, mivel csak megvalósításában tér el a fent felsoroltak egyik

Óra (Clock)	
	AZ SC algoritmus eredményeit állítja elő, viszont itt a lapok egy FIFO lista helyett egy körkörös listára kerülnek felfűzésre a betöltés sorrendje szerint, ahol egy mutató jelöli a legrégebbit. Az algoritmus az R bitet vizsgálva dönt a kivitelről, ha be van állítva, törli és tovább lép, különben kiválasztja kivitelre a lapot.

A feladatokhoz kapcsolódó metrikák és statisztikák

Az alábbiakban témakörönként összefoglalásra kerülnek a megvizsgált metrikák és egyéb adatok.

A taszkütemezéshez kapcsolódó metrikák

Multiprogramozás foka	
[darab]	Megadja a rendszerben adott időpillanatban levő megkezdett, de be nem fejezett folyamatok számát.
Központi egység kihasználtság (CPU Utilization)	
[%]	Meghatározza, hogy az összesen felhasznált CPU idő hány százaléka lett felhasználva ténylegesen hasznos munka végzésére, azaz folyamatok futtatására. Ezt az értéket csökkenti, ha a CPU adminisztrációs, ütemezési stb. feladatokat lát el, vagy éppen semmit sem csinál.
	$\frac{\sum t_{CPU\ hasznos\ munkával\ töltöt\ idő}}{\sum t_{CPU\ összes\ tevékenységgel\ eltöltött\ idő}} * 100[\%]$
Átbocsátó képesség (Throughput)	
[1/s]	Meghatározza az elvégzett munkák számát egy időegységre nézve.
	$\frac{elvégzett\ munkák\ száma}{idő}$
Várakozási idő (Waiting time)	
[s]	Meghatározza azt az időt, amelyet a feladat várakozással tölt.
	$\sum (t_{várakozó} + t_{futásra\ kész} + t_{felfüggesztett} + t_{hosszú\ távú\ ütemezéséig\ eltelt})$
Körülfordulási idő (Turnaround time)	
[s]	Meghatározza, hogy a feladat rendszerbe helyezésétől számított, teljesítésig eltelt időt

	$\sum(t_{végrehajtási\ idő} + t_{várakozási\ idő})$

Az alábbiak modellezési okok miatt nem kerültek felhasználásra:

Válaszidő (Response time)	
[s]	Meghatározza a feladat elkezdésétől a rendszer első érzékelhető reakciójáig mért időt. Tehát az az idő, amit a rendszer az első kimenet előállításáig eltölt.
	Számítási mód: a metrika kiszámítása a folyamatok által produkált első kimenet idejét igényelné, ezáltal viszont körülményes lenne a bemenő feladatok megadása. A feladat megoldásában konkrét kódvégrehajtás hiányában megvalósítása nem lehetséges.

A memóriafoglaláshoz kapcsolódó metrikák

Memória kihasználtsága	
[%]	Meghatározza, hogy a memória területeinek hány százaléka van lefoglalva a folyamatok által
	$\frac{\sum foglalt\ területű\ partíció\ mérete}{\sum partíció\ mérete} * 100[\%]$
	Megjegyzés: belőle könnyen számolható a kihasználatlan terület aránya az összterülethez képest a 100%-ból való kivonással
Kihasznált és kihasználatlan területek aránya	
[arány]	Meghatározza a kihasznált és kihasználatlan területek arányát
	$\frac{\sum lefoglalt\ területet\ tartalmazó\ partíciók\ mérete}{\sum nem\ lefoglalt\ területet\ tartalmazó\ partíciók\ mérete}$
Kihasznált partíciók száma	
[darab]	Meghatározza, hogy a memóriában levő összes partíció közül hány van kihasználva
Kihasználatlan partíciók száma	

[darab]	Meghatározza, hogy a memóriában levő összes partíció közül hány kihasználatlan, azaz olyan szabad terület ahova folyamat még nem foglalt partíciót
Kihasznált partíciók aránya az összes partícióra nézve	
[arány]	Meghatározza a folyamatokhoz tartozó partíciók arányát az összes partícióhoz képest
	$\frac{\sum kihasznált\ partíciók\ száma}{\sum partíció\ száma} * 100[\%]$
	Megjegyzés: belőle könnyen számolható a kihasználatlan partíciók aránya a 100%-ból való kivonással
Kihasznált partíciók aránya az összes partícióra nézve	
[arány]	Meghatározza a folyamatokhoz tartozó partíciók arányát az összes partícióhoz képest
	$\frac{\sum kihasznált\ partíciók\ száma}{\sum partíció\ száma} * 100[\%]$
	Megjegyzés: belőle könnyen számolható a kihasználatlan partíciók aránya a 100%-ból való kivonással
Kihasznált és kihasználatlan partíciók aránya	
[arány]	Meghatározza a folyamatokhoz tartozó partíciók és a szabad területek arányát
	$\frac{\sum kihasznált\ partíciók\ száma}{\sum kihasználatlan\ partíciók\ száma}$

A lapcsere algoritmusokhoz kapcsolódó metrikák

Laptábla minimális mérete

[byte]	A logikai címzést végző bitekből meghatározott minimális táblaméret közvetlen leképezést feltételezve (a tényleges méret ennél nagyobb lehet, különböző jelzőbitek kerülhetnek tárolásra stb.)
	Számítási mód: a logikai címek adott laptábla-szintre vonatkozó bitsorozatának és a bejegyzések számának szorzatából adódó méret
Lapok mérete	
[byte]	A lapon belüli eltolást végző bitekből meghatározott lapméret
	Számítási mód: a lapon belüli eltolásra használt bitekből számított maximális érték, egy byte-nyi natív szóméretet feltételezve
Átlagos címleképezési idő	
[ns]	Meghatározza, hogy a címleképezések mennyi idő alatt kerültek végrehajtásra
	Számítási mód: átlaga a laptáblához fordulási és az asszociatív memóriához fordulási idők minimumainak
	Megjegyzés: nagyszámú memória hozzáférés idejének átlagolásával az értéke tart az effektív hozzáférési időhöz

Laphibák száma	
[darab]	Meghatározza, hogy az algoritmus futása során hány laphiba keletkezett
	$\sum laphibák\ száma$
Átlagos címleképezési idő	
[ns]	Számítási mód: az egyes memória hozzáférési idők és a laphiba kiszolgálási idők elemeiből alkotott sorozatnak és a címleképezések számának hányadosa
	Megjegyzés: nagyszámú lap hozzáférés idejének átlagolásával az értéke tart az effektív hozzáférési időhöz

A következő metrikák modellezési okok miatt nem kerültek megvalósításra:

Munkahalmaz mérete	
Dinamikus, időben változó fogalom, mely meghatározza a folyamat azon lapjainak halmazát (egyúttal méretét is), melyet egy meghatározott időintervallumban (munkahalmaz ablak) a futás során igénybe vesz.	
Laphiba gyakoriság	**(PFF: Page Fault Frequency)**
Meghatározza, hogy a folyamat futása során rögzített számú tárban helyet kapó lapkészlet mellett, mi a laphibák előfordulásának várható értéke. Ez az érték a memórián belüli címtartomány nagyságának függvénye.	
Laphibák között eltelt idő	**(Interfault Time)**
A laphiba gyakorisággal egyenértékű, két laphiba között eltelt idő mérésén alapuló metrika. A laphiba gyakorisággal ellentétben, ennek mérése könnyen megvalósítható	

A bemenetek megadásának leírása

Az alábbiakban összefoglalásra kerülnek a további bemenetek gyártásához szükséges információk.

Az ütemezési feladatok statikus leíró modellje

A feladat ütemezési algoritmusokat megvalósító megoldás a következő bemenetet leíró modellel rendelkezik:

- **CPU ütemezési algoritmus neve** – az ütemezést végző algoritmus megnevezése (FIFO, RR, SJF, SRTF)
    - **Ütemezési algoritmus esetleges paraméterei** – az algoritmusokhoz szükséges paraméterek megadása, például időszelet nagysága
- **Az i. erőforrás típus neve** – a folyamat által lefoglalható bármilyen erőforrás, be- és kimeneti egység, hardverelem, logikai erőforrás stb. absztrakt megnevezése.
    - **Erőforrás készlet** – a megnevezett típusból rendelkezésre álló erőforrások száma.
- **Az i. folyamat neve** – az ütemezni kívánt folyamat absztrakt elnevezése.
    - **Folyamat beérkezési ideje** – a folyamat rendszerbe kerülésének ideje

- **Folyamat CPU és IO löketidőinek sorozata** – a futást jellemző erőforrás terheltség idők megadása. Elsőként mindenképpen egy CPU löketidő kerül megadásra, csak ez után következhetnek a perifériák használatára jellemző IO löketidők. A löketidők CPU és IO löketidők a bemenő sorozatban mindig váltakozva fordulnak elő, azonos típusú löketidő kétszer egymás után történő deklarálása nem értelmes. Az időtartamok ms-ban kerülnek megadásra.
  - **CPU löketidő** – a folyamat processzor igényének löketideje. Legalább egy processzorlöket megadása mindenképpen szükséges.
  - **Szinkron IO löketidő az erőforrás megjelölésével** – a folyamat által használt, szinkron módon igényelt, azaz blokkoló, be- és kimeneti egységhez tartozó IO löketidő. Feltételezve, hogy van megelőző CPU löket, mely során az egység felprogramozásra került.
  - **Aszinkron IO löketidő az erőforrás megjelölésével** –a folyamat által használt, aszinkron módon igényelt, nem blokkoló, be- és kimeneti egységhez tartozó IO löketidő. Feltételezve, hogy van megelőző CPU löket, mely során az egység felprogramozásra került.

- **Átlagos adminisztrációs idő** – az operációs rendszer futásának átlagos ideje az ütemezések, kontextus váltások, készülékek elérésének stb. modellezéséhez. Az időtartam ezred ms-ban kerül megadásra.

A memóriafoglalási feladatok statikus leíró modellje

A memóriafoglalás bemutatására szánt modell a következő paraméterezhetőségi lehetőségekkel rendelkezik:

- **Memória allokációját végző algoritmus megnevezése** – az allokációt végző algoritmus megnevezése (FF, NF, BF, WF)
- **Fizikai tár mérete** – az operációs rendszer és a folyamatok által igénybe vehető fizikai memória terület. Gyakorlatban a mérete 2 hatványok összege, itt azonban ez nem kerül ellenőrzésre. KByte-okban kerül megadásra.
  - **Rendszer memória terület mérete** – az operációs rendszer által igénybe vett, folyamatok által nem használható terület. Mérete Kbyte-okban megadva.

- **Felhasználói memória terület mérete** – a folyamatok által virtuális memóriakezelésen keresztül igénybe vehető memóriaterület. Mérete Kbyte-okban megadva.

- **Memória foglaltságának kiinduló állapota** – meghatározza, hogy kezdetben hány változó méretű partíció van a rendszerben, azok milyen méretűek. Továbbá információval rendelkezik arról, hogy e területek szabadak-e, vagy valamilyen folyamat már használatba vette-e őket.

- **Beérkező memóriafoglalási és felszabadítási igények sorozata** – a folyamatok által az operációs rendszerhez beérkező memóriafoglalási és memória felszabadítási igényeket tartalmazza idő szerinti sorrendben. A lefoglalandó terület nagysága Kbyte-okban kerül megadásra. A folyamatok által lefoglalt területek a partíció azonosítójának megnevezésével felszabadíthatók.

A virtuális memória címleképezés feladatok statikus leíró modellje

A címleképezést egy folyamat által bemutató statikus modell paraméterezéséhez a következő bemeneti értékek megadására van lehetőség:

- **Címzésre használt bitek száma** – az értéket a valós rendszerekben a processzorok címzési módja határozza meg. Az itt megadható értékek a 8 és 128 közé eső 2 hatványok értéke

    - **A laptáblán használt logikai címbitek száma** – megadja, hogy a címzésre használt bitekből mennyi kerül felhasználásra a laptáblában levő lapok címzésére, egyben meghatározza a laptáblában levő lapok maximális számát is.

    - **Memória nagysága** – a benne tárolható lapok számának megadásával

    - **Memória hozzáférési idő** – az effektív hozzáférési idő metrikának számításához szükséges. Az időtartam ns-ban kerül megadásra

    - **Asszociatív tár megléte** – megadható, hogy van-e a rendszerben a címleképezés gyorsító táraként funkcionáló asszociatív memória.

    - **Asszociatív tár nagysága** – megadható, hogy a gyorsító tárként használatos asszociatív memória hány címet képes tárolni

    - **Asszociatív tár hozzáférési idő** – az effektív hozzáférési idő metrikának számításához szükséges. Az időtartam ns-ban kerül megadásra
    - **Folyamat címtartományának mérete** – a folyamat logikai memóriája által foglalt címtartomány, amely a végrehajtáskor a lapméreteknek megfelelő blokkokra darabolódik. Egy része a fizikai memóriában, másik része a háttértáron kerül tárolásra. Megadható a folyamat utolsó értelmes címével.

- **Memória terület foglaltságának kialakítása** – megadható, hogy a tár (pillanatnyi, végig statikus) tartalma automatikusan, egy megnevezett módszer által, vagy manuálisan kerüljön megadásra. A tárban helyet kaphatnak a vizsgált, vagy a nem vizsgált folyamatok lapjai, valamint szabad memóriakeretek.
- **Asszociatív tár foglaltságának kialakítása** – megadható a lapok sorszámainak megadásával
- **A bemenő események sorozata**
    - **Lap memóriába töltése** – megadható a lap sorszámának és a keret sorszámának megadásával
    - **Keret memóriából törlése** – megadható a keret sorszámának megadásával
    - **Lapcím asszociatív memóriába töltése** – megadható a lap sorszámának megadásával
    - **Lapcím asszociatív memóriából való törlése** – megadható a lap sorszámának megadásával
    - **Folyamat által hivatkozott virtuális címek feloldása** – a folyamat végrehajtása során a leképezni kívánt virtuális címek. Értékkészletének a folyamat logikai címtartományába kell esnie.

A lapcsere algoritmus feladatok statikus leíró modellje

A lapcsere algoritmusok bemutatásának valós idejű motorjaként működő modell paraméterezéséhez a következő bemeneti információk megadására van szükség:

- **Alkalmazott algoritmus megnevezése** – az áldozat lap kiválasztását végző algoritmus megnevezése (OPT, FIFO, SC, LFU, LRU, NRU)
    - **Az algoritmus esetleges paramétere**

- **Vizsgált folyamat által felhasználható lapok száma** – a folyamat maximálisan a megadott számú lapot tarthatja a memóriában
- **Bemeneti események sorozata** – olyan események sorrendje, amelyek egy virtuális tárkezelésen alapuló rendszerben felmerülhetnek. Ahhoz, hogy a szimuláció során bizonyos eseményeket modellezni tudjuk, a feladat egyszerűsítései miatt ezek bemenő információ formájában kerülnek megadásra. Ilyen információk az alábbi pontokban meghatározottak lehetnek:
    - **Laphivatkozások** – ezáltal szimulálható a valós folyamatok kódjában virtuális címleképezés által előálló hivatkozott lapok sorrendje
    - **Jelzőbit beállítások sorrendje** – ezek alakulását a valós rendszerekben a folyamat kódjának végrehajtása, valamint az operációs rendszer működésének együttese futás időben befolyásolja. Mivel a demonstráció során nincsenek konkrét végrehajtandó utasítások, így ilyen működést a szimuláció során csak bemeneti információ formájában, előre definiált sorozat megadásával kerül megadásra
        - **R bit beállítása vagy törlése** – a valós kódban hivatkozott lapok szimulációjához
        - **M bit beállítása vagy törlése** – a valós kód által módosított lapok szimulációjához
        - **R bitek törlésének periódus eseménye** – bizonyos algoritmusok futásához szükséges esemény (mely a pontosabb modellezés végett nem lépés periódusként, hanem eseményként definiált)
- **Memória hozzáférési idő** – az utasítások végrehajtása során a virtuális címek feloldásának átlagos ideje, amennyiben a lap a memóriában tartózkodik. Az effektív hozzáférési idő metrikának számításához szükséges. Az időtartam ns-ban kerül megadásra.
- **Laphiba kiszolgálási idő** – az utasítások végrehajtásakor a virtuális címek feloldása során a memóriában nem tartózkodó lapok behozataláig eltelt átlagos idő. Mivel az effektív hozzáférési idő metrikához szükséges, ezért itt azzal a feltételezéssel élünk, hogy a folyamat a lapbehozatal után azonnal folytatja futását és nem kell megvárni, amíg az ütemező futásra kiválasztja. Ha ez a szemlélet nem felelne meg nekünk,

tekintsünk erre az időre úgy, mint az átlagos lapkiszolgálási időre, melybe beleszámolandó az átlagos folyamat beütemezési idő is. Az időtartam ns-ban kerül megadásra

Modellbővítési lehetőségek

Az alábbiakban összegzésre kerülnek a modell bővítésének lehetőségei.

Az ütemezési algoritmusok megoldásának modellbővítési lehetőségei

Mivel a folyamatok ütemezésének bemutatása során a valós rendszerek megoldásaihoz képest sok egyszerűsítés került alkalmazásra, ezért kiegészítési lehetőségként szintén sok lehetőség adódik.

Pontosabb időviszonyokat térképezhetünk fel, ha a szimuláció során nemcsak egy átlagos adminisztrációs időt használunk, hanem foglalkozunk az ütemezési, adminisztrációs, megszakításkezelési, periféria készenléti stb. idők megadásával is. Még pontosabb képhez juthatunk, ha időtartományok megadása helyett a folyamatainkat absztrakt utasításokkal hozzuk létre és ezek végrehajtását szimuláljuk virtuális erőforrásainkon. Bonyolíthatjuk a helyzetet, ha erőforrásaink használata során nemcsak az egyszerűség okán alkalmazott FIFO lista megoldást alkalmazzuk, hanem a valós rendszerekben előfordulókat is szimuláljuk. A használható perifériák típusa is szerteágazó, a pontosabb idők feltérképezéséhez és bemutatásához egyáltalán nem szükséges az általam használt egyszerűsítésekkel élni. Egyeseknél feltételezhetjük a processzor intenzív használatát is a periféria használata mellett. A bonyolultság növelésének újabb dimenziója nyitható meg a multiprocesszáló rendszereken történő multiprogramozottság bemutatásával, többprocesszoros rendszerek, magok alkalmazásával. Összetettebb rendszerarchitektúrákban specifikus ütemezési feladatok megvalósítására is szükség lehet. Könnyen előfordulhat, hogy szűkös erőforráskészletünk egyidejű használatának mikéntjéről kell döntenünk.

A memóriafoglalás megoldásának modellbővítési lehetőségei

Mivel a memóriafoglalás szerteágazó terület és eme megvalósítás elkészítésekor sok egyszerűsítést alkalmaztam, ezért ebből a megvalósításból kimaradó megoldások bármelyikét lehetne implementálni memóriafoglalás címszó alatt. Hasznos lenne például az elkészített változó méretű partíciók lefoglalását például úgy kiterjeszteni, hogy a folyamatok csak megadott kezdőcímekre tölthetők be a memóriában, vagy egyéb

címfordítási megkötéseknek kell eleget tenniük; az ilyen megkötésekkel történő szimuláció érdekes szituációkhoz vezethet. Esetleg adott időközönként szemétgyűjtőt futtatni nagy lefutási idővel és megvizsgálni, hogyan tömbösíti a memóriában a lefoglalt területeket és hogyan képez egységes területet a szabad területekből. Hasznos lehet megfontolni a rögzített méretű partíciók alkalmazását, hogy látható legyen a belső tördelődés, esetleg folyamatütemezővel kiegészíteni, és a várakozási sor(ok) tartalmának időbeli változását vizsgálni. További szimulációs lehetőségek rejlenek a folyamatok késleltetett betöltésének módszereiben, mint például a dinamikus betöltés, dinamikus könyvtárbetöltés és az átfedő programrészek technikája. Megfontolandó, hogy szimuláció során átlapolt tárcserét hajtsunk végre az ütemező által kiválasztott folyamatokon. Lap-, szegmensszervezésű, vagy egyéb hierarchikus vagy kombinált megoldásokat alakíthatunk ki, és ezeken végezhetünk szimulációkat. (Lapszervezésű megoldások egy része egy későbbi alfejezetben kerülnek kidolgozásra). Az előzőek bármelyikével bővíthetjük a memóriafoglalás címszó alá tartozó demonstrációs eszközök tárházát.

A virtuális memória címleképezés megoldásának modellbővítési lehetőségei

Az előző részfeladatokhoz képest ez a feladatrész a statikusságából adódóan kevés kiterjesztési ponttal rendelkezik. A címleképezés bemutatása során ugyanis nem igazán van mód új lehetőségek, algoritmusok bemutatására, ennek oka főleg a feladat speciális jellege. Egyetlen kiterjesztési pontként a feladatban opcionálisan alkalmazott asszociatív tár tartalmának frissítését tudom elképzelni. A feladat megvalósítása során csak a „buta”, időbeli és térbeli lokalitáson alapuló megoldás kerül implementálásra. Ennél azonban jóval fejlettebb algoritmusok is alkalmazhatóak lennének a címfeloldást segítő asszociatív tár frissítésére, melyekkel jelen szakdolgozat keretében nem kívánok foglalkozni, mivel nem képezik a feladat központi részét. További kiterjesztési pontként a felhasználó számára lehetőséget biztosítok a memória tartalmának inicializálására. Ezáltal ha a megvalósításra kerülő alap stratégiák (mint pl. a vizsgált és a nem vizsgált folyamatok tárigényeinek véletlenszerű, sorrendi, esetleg manuális módon történő megadása) a felhasználó számára nem elégségesek, akkor lehetősége nyílik további stratégiák implementálására (mint pl. a különböző folyamatok számára egy eloszlásfüggvény mentén történő tárkihasználtság beállítása stb.).

Felmerülhet az igény a virtuális memória címleképezésének bemutatására a memória kezdeti allokációjától kezdve egészen a lapok tárcseréjének bemutatásáig, dinamikusan alakuló memóriatartalom mellett, több folyamat futásának, egymásra hatásának szemléltetésével. Ebben az esetben eme feladat kiterjesztési pontokon keresztül történő bővítése nem elegendő. Ekkor ugyanis egy teljes szimulációs környezet felépítésére lenne szükség (absztrakt kód végrehajtása, folyamatütemezési megvalósítás, tárallokáció, lapcserék, tárcserék stb.) melynek az itt létrehozásra kerülő modell csak egy részelemét alkotná. A jelenlegi kiterjesztési pontokon keresztül egy ilyen szimulációs környezet kialakítása jelenleg nem támogatott, de kiegészítésképpen könnyen el lehetne képzelni egy ilyen eshetőséget is. Ez esetben érdemes megfontolni, hogy a bonyolultság ilyen mértékű növelésével nem veszítünk-e demonstrációs eszközünk hatékonyságából.

A későbbiek folyamán, amennyiben igény mutatkozna rá, érdemes lehet a virtuális tárkezelési megvalósítások sorának bővítésével foglalkozni. Bemutathatóak lehetnének például többszintű laptáblák vagy szegmensszervezésű megoldások is. Érdemes lehet vizualizálni a szegmenseken történő túlcímzés esetén keletkező hibamegszakítások okait. Kombinált szegmens- és lapszervezésű megoldások is könnyen vizualizálhatóak. Bemutatható például a logikai cím hármasának felbontása és értelmezése, vagy esetleg az, hogy ezt a többlépcsős folyamatot miképpen gyorsítják az asszociatív tárak.

A lapcsere algoritmusok megoldásának modellbővítési lehetőségei

Mivel a lapcsere algoritmusok használatához kapcsolódóan, itt csak a legelemibb részek kerülnek bemutatásra, ezért a későbbiek folyamán megfontolandó a feladat további részekkel történő kiegészítése. Bár a megvalósítás során az egyetlen előre tervezett kiterjesztési pont egy újabb algoritmusnak felvétele és működésének bemutatása, az alkalmazás több irányba is kiegészíthető.

A fenti feladat csupán a lokális tárgazdálkodás lapcsere algoritmusainak bemutatására lett felkészítve egy folyamat lapigényeinek kiszolgálása által, azonban a megoldás kiegészítésével a meglevő algoritmusok alkalmazhatóak a globális tárgazdálkodás bemutatására is. Ehhez nem csupán egy folyamat végrehajtása során keletkező lapigények sorozatát kellene alapul venni, hanem számolni kellene más feladatok lapigényeivel is. Még pontosabb szimulációt lehet kialakítani, ha a lapigények

sorozatának feldolgozása helyett egyfajta szimulált kódvégrehajtás kerül megvalósításra. Ekkor viszont érdemesebb a virtuális címleképezést bemutató megvalósítással egybekapcsolni a megoldást. Ez nagyban növelné a bonyolultságot, csökkentené a szemléltethetőséget, nem beszélve a bemenő információk méretének megnövekedéséről, így viszont lehetőséget adna olyan metrikák számítására (mint pl. munkahalmaz mérete, laphiba gyakoriság, laphibák között eltelt idő), amelyekre itt nem volt lehetőség. Szintén bemutathatók lennének a dinamikus tárgazdálkodás alapelvei, hogy a metrikák mérésével az algoritmusok miképpen módosítják az egyes folyamatokhoz tartozó lapkészleteket, és hogy teljesítmény szempontjából az egyes tartományokba eső (lapbőségben levő, lap szűkében levő, elfogadható tartományban levő) folyamatok mikor kerülnek átsorolásra más tartományokra. Mivel minden folyamat futása során eltérő memóriaigénnyel rendelkezik, a valós körülmények között fennálló állapotoknak megfelelő szimulációs információk megadása körülményessé teheti a bemeneti feladatok kialakítását. Szintén a valós időviszonyok vizsgálatával lenne érdemes megvalósítani a fenti algoritmusok működésének bemutatását egy áldozatlista karbantartása mellett, melynek célja gyorsítás elérése. Ez a lista a CPU szabad idejében kerülne kialakításra. Komplex kialakítás során egy operációs rendszer által periodikusan felébresztett terhelésmentesítést végző háttérfolyamat futása is elképzelhető. Ez a folyamat a háttértár szabad idejében viszi és hozza ki/be a szükséges lapokat, ezáltal szintén gyorsulás érhető el vele. Fokozható a gyorsulás az áldozatlista és a paging daemon együttes használatával.