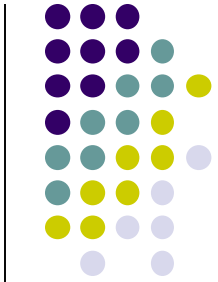
1. Blokea:
Sistema Eragileak

Informazio-Sistemen Arkitektura

Telekomunikazio Teknologiaren Ingeniaritzako Gradua (3. Maila)

TELEK:O
UPV/EHU Bilbao

1. Blokea - Edukiak

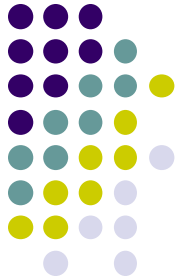


1. Ordenagailuen Arkitekturako kontzeptuak
2. Sistema Eragileak. Sarrera.
3. Prozesuak
   - Prozesuak eta hariak
   - Prozesuen arteko komunikazioa
   - Komunikazio eta Sinkronizazio Mekanismoak
   - Planifikazioa
4. Sarrera/Irteera
5. Memoriaren Kudeaketa
   - Memoria birtuala edo alegiazko memoria
   - Orrikapena eta Segmentazioa
   - Ordezkapeneko algoritmoak
6. Fitxategi Sistema
   - Fitxategiak eta direktorioak
   - Fitxategi Sistemaren antolaketa

5. Gaia – Edukiak
MEMORIAREN KUDEAKETA

1. Memoria kudeaketaren helburuak
2. Exekutagarrien formatua eta eskualdeak
3. Ondoz ondoko memoria-esleipena
4. Swapping eta memoria birtuala
5. Orrikapena
6. Segmentazioa
7. Segmentazio orrikatua
8. Orrikapena antolatzeko algoritmoak
9. Memorian proiektatutako fitxategiak

5. MEMORIAREN KUDEAKETA
5.1 HELBURUAK

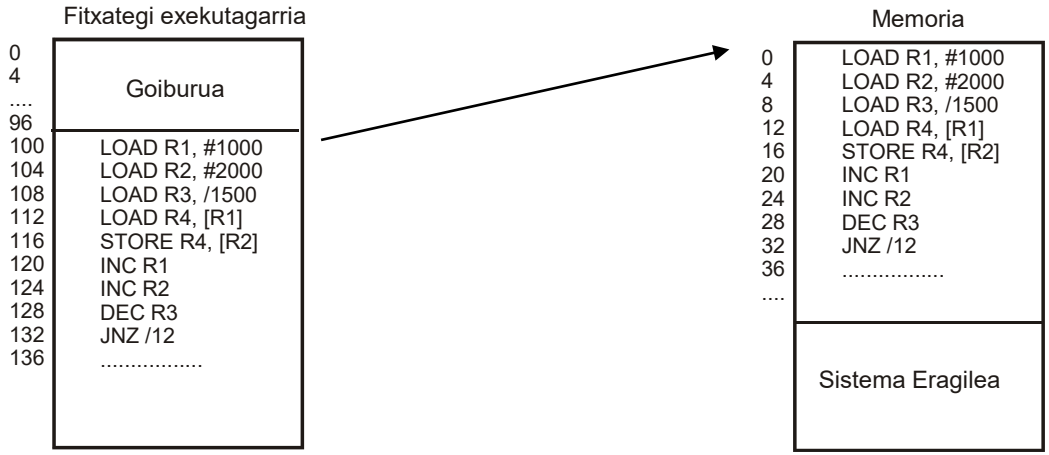


- SE-ak baliabideak multiplexatzen ditu prozesuen artean
  - Prozesu bakoitzak makina guztia beretzat balitz bezala ikusten du
  - Prozesuen kudeaketa: Prozesadorea prozesuen artean banatu
  - Memoriaren kudeaketa: Memoria banatu

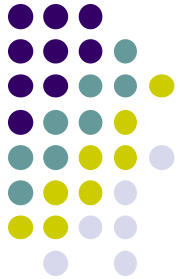
- Helburuak:
  - Prozesu bakoitzari espazio logiko independentea eskaini
  - Prozesuen artean babesa bermatu
  - Prozesuen artean memoria partekatzea ahalbidetu
  - Prozesuaren eskualdeei euskarria eman
  - Sistemaren errendimendua maximizatu
  - Prozesuei memoria-mapa oso handiak eskaini

5. MEMORIAREN KUDEAKETA
5.1 HELBURUAK

- **Espazio logiko independenteak**
  - Ezin daiteke aldez aurretik jakin non kokatuko den programa bat sistemaren memorian
    - Exekuzio desberdinetan ere alda daiteke
  - Makina-kodeak 0 eta N arteko erreferentziak sortzen ditu
    - Adibidea: Programa batek, 1000 helbidetik abiatuta gordetzen den bektore bat, 2000 helbidetik aurrera kopiatzen du. Bektorearen luzera 1500 helbidean dago.
      - Monoprogramazioan eta SE-a helbide gorenetan kargatuta badago:



5. MEMORIAREN KUDEAKETA
5.1 HELBURUAK



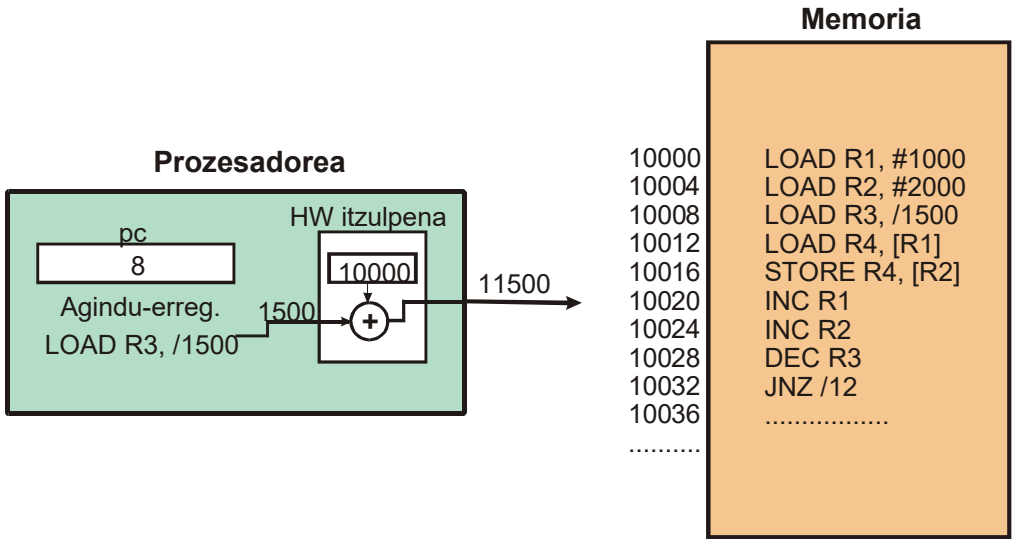
- **Birkokapena**:
  - Multiprogramazioa duten SE-etan derrigorrezkoa:
    - Helbide logikoak: programak sortutako helbideak
    - Helbide fisikoak: prozesuari esleitutako memoria nagusiko helbideak
  - Itzulpenaren helburua: *Itzulpena(ProzID, hel_log)* $\rightarrow$ *hel_fis*

  - Birkokapenak prozesuaren espazio logiko independentea sortzen du
    - SE-a prozesuen espazio logikoa atzitzeko gai izan behar da

  - Bi alternatiba:
    - Hardware edo Software birkokapena

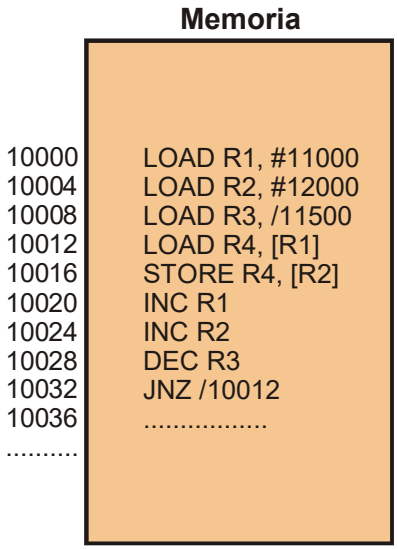
5. MEMORIAREN KUDEAKETA
5.1 HELBURUAK

- **Hardware birkokapena**: MMU-ak (HW) itzulpena egiten du
  - SE-aren ardurak:
    - Prozesu bakoitzaren itzulpen-funtzioa gorde
      - Prozesu desberdinek itzulpen-funtzio desberdina
    - Hardwareari zein funtzio erabili agindu
  - Exekutagarriak diren moduan kargatzen dira RAMean

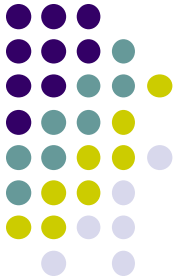


5. MEMORIAREN KUDEAKETA
5.1 HELBURUAK

- **Software birkokapena**:
  - Itzulpena programaren kargaren zehar egiten da
    - Helbideak eta datuak bereiziz, eta ondorioz:
  - RAMean dagoen programa eta exekutagarriak desberdinak
  - Desabantailak:
    - Ez du segurtasuna bermatzen
    - Programaren kokapena ezin daiteke aldatu exekuzioan zehar



5. MEMORIAREN KUDEAKETA
5.1 HELBURUAK



- Prozesuen artean **babesa bermatu**
  - Monoprogramazioan: SE-a prozesutik babestu
  - Multiprogramazioan: Gainera, beste prozesuetatik

  - Itzulpenak espazio disjuntuak sortu behar ditu

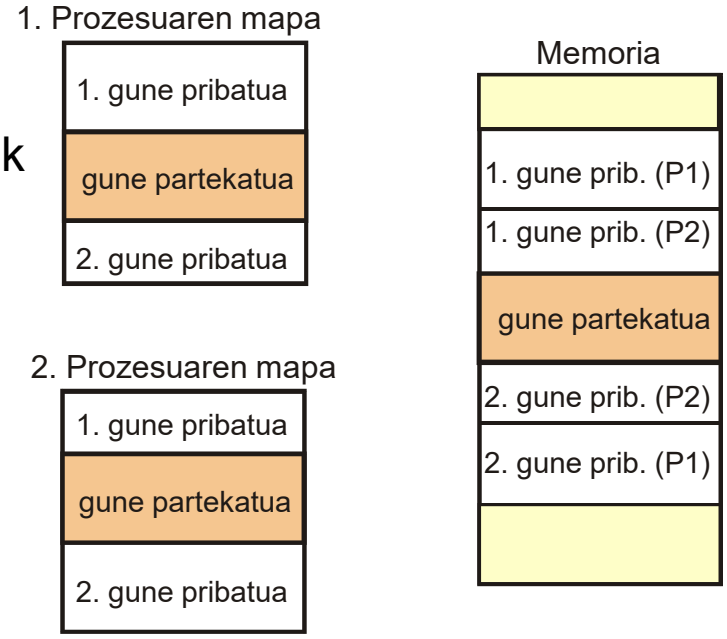
  - Programak sortzen dituen helbide guztiak balidatu beharra dago:
    - Detekzioa prozesadoreko hardwareak egiten du (Hardware kokapena suposatzen da)
    - Tratamendua SE-ak egingo du

  - S/I gailuek memoria dutenean (adb. Bideo-txartelak) :
    - Itzulpenak ukatu egingo die prozesuei S/I gailuetako helbideak sortzen

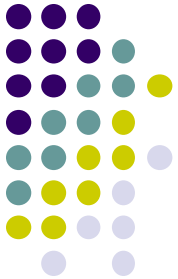
5. MEMORIAREN KUDEAKETA
5.1 HELBURUAK

Babesa bermatzearen kontra

- Prozesuen artean **memoria partekatzea**
  - 2 prozesu edo gehiagoren helbide logikoei helbide fisiko bera dagokie
  - SE-aren kontrolpean
  - Onurak:
    - Prozesu beraren instantzia desberdinek kode bera partekatu
    - Prozesuen arteko komunikazio-mekanismo azkarra
  - Memoria-esleipena ezin daiteke izan ondoz ondokoa
  - Erakusleen beharrizana



5. MEMORIAREN KUDEAKETA
5.1 HELBURUAK

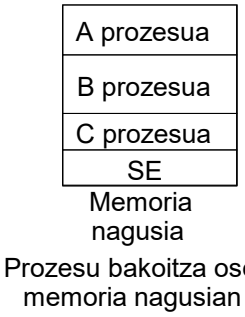


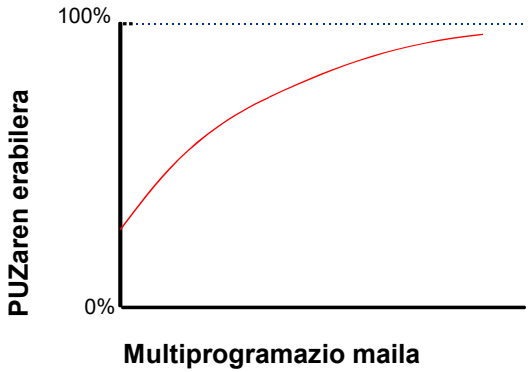
- **Prozesuaren eskualdeei euskarria eman**
  - Prozesuen mapa ez da homogeneoa
    - Eskualde bakoitzak bere ezaugarri propioak ditu
    - Adibidea: Kode-eskualdea exekutagarria da baina ezin da idatzi
  - Prozesuaren mapa dinamikoa da
    - Eskualdeen luzera aldatuz doa (adib. pila edo stack)
    - Eskualdeak sortu eta desegiten dira
    - Esleitu gabeko eskualdeak (hutsuneak) daude
  - Memoria kudeatzaileak hurrengo premiak bete behar ditu:
    - Eskualdeetara debekatutako atzipenak antzeman
    - Hutsunetara eginiko atzipenak antzeman
    - Hutsuneetarako memoria-erreserbak ekidin

    ⇨ Itzulpen funtzio aldaketa
  - SE-ak prozesu bakoitzaren eskualdeen taula gorde behar du
    - Ze eskualde eta ze tamainakoak

5. MEMORIAREN KUDEAKETA
5.1 HELBURUAK

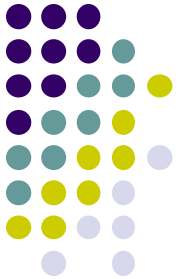
- Sistemaren **errendimendua maximizatu**
  - Memoria banatu **multiprogramazio-maila** maximizatzeko
    - Multip. mailak ↑→ prest egoeran egoteko probabilitateak ↑
    - Multip. mailak ↑→ memoriaren premiak ↑
    - Memoria birtualik gabeko sistema bateko mugak
      - RAMaren luzera
      - Prozesuen luzera





    - Memoria birtuala duten sistemetan, multiprogramazio-maila ez dago hain mugatuta RAMaren eta prozesuen luzeragatik

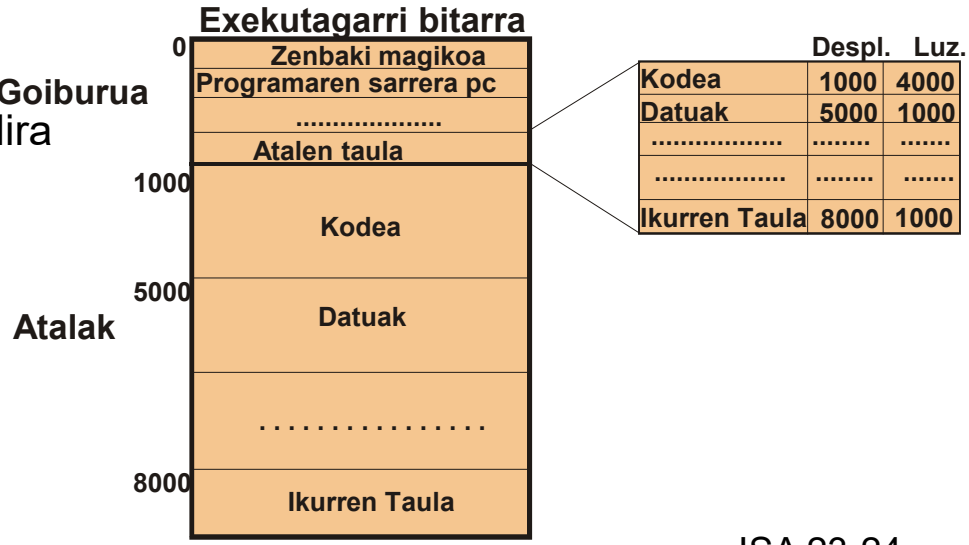
5. MEMORIAREN KUDEAKETA
5.1 HELBURUAK



- Prozesuei **memoria-mapa oso handiak eskaini**
  - Prozesuek gero eta memoria-mapa handiagoa behar dute
    - Programazio lengoaiak eta teknikak, OOP adibidez
    - Aplikazioak gero eta ahaltsuagoak eta berritzaileagoak
  - Memoria birtualari esker konponduta
  - Antzina *overlay*-ak erabiltzen ziren:
    - Programa estekatzerakoan modulu-multzoak aukeratu
    - Exekuzioan zehar memorian dauden multzoak aldatuz doaz
    - Multzoen lana amaitu ondoren hurrengo multzoak kargatzen dira
    - Ez da gardena: Programatzailearen intuizioaren menpeko eraginkortasuna

5. MEMORIAREN KUDEAKETA
5.2 EXEKUTAGARRIEN ESKUALDEAK

- Bitarraren egitura SE-aren araberakoa
  - Linuxen adib.: *Executable and Linkable Format (ELF)* zenbaki magikoa aurkituko dugu goiburuetan
  - Egitura orokorra: Goiburua eta atalak edo eskualdeak
  - Goiburuan: Exekutagarria interpretatu ahal izateko beharrezko kontrolezko informazioa
    - Exekutagarria dela identifikatzeko zenbaki magikoa
    - Programaren sarrera puntua (hasierako pc)
    - Eskualdeen taula
      - Hasierakoak
      - Exekuzioan berriak sortuko dira



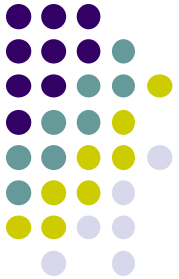
5. MEMORIAREN KUDEAKETA
5.2 EXEKUTAGARRIEN ESKUALDEAK

- Atal motak:
  - Kodea (Testua): Programaren kodea
  - Datuak: Aldagai orokorrak
    - Baliodunak
    - Balio gabeak: eskualdeen taulan agertzen da bere deskribapena, baina ez da exekutagarrian gordetzen.
  - Pila edo Stack:
    - Zergatik ez dago aldagai lokalen edo funtzioen parametroen atalik bitarrean?
  - Heap: Memoria dinamikoa (malloc(), free())
  - Memorian proiektatutako fitxategiak
  - Partekatutako memoria
  - Liburutegi dinamikoak (Testu+Datuak)
  - Harien pilak
  - Arazketarako Ikurren Taula
- Prozesuaren Irudia: Eskualdeen batuketa

Prozesuaren irudia

HUTSUNEAK

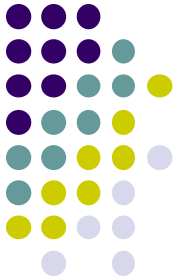
5. MEMORIAREN KUDEAKETA
5.2 EXEKUTAGARRIEN ESKUALDEAK



- Eskualdeen ezaugarriak:

Eskualdea	Euskarria	Babesa	Partek/Prib	Luzera
Kodea	Fitxategia	RX	Partekatua	Finkoa
Datuak h.b.	Fitxategia	RW	Pribatua	Finkoa
Datu h.b.g.	Euskarri gb.	RW	Pribatua	Finkoa
Pilak	Euskarri gb.	RW	Pribatua	Aldakorra
Heap	Euskarri gb.	RW	Pribatua	Aldakorra
Proiekt. F	Fitxategia	Erabiltzaileak	Partek./Prib.	Aldakorra
Partek. M.	Euskarri gb.	Erabiltzaileak	Partekatua	Aldakorra

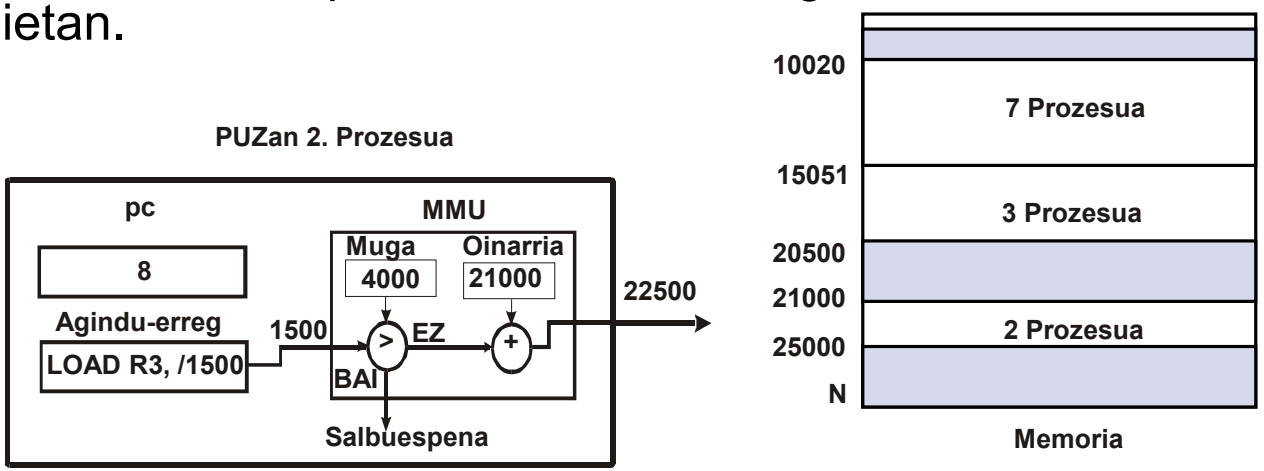
5. MEMORIAREN KUDEAKETA
5.2 EXEKUTAGARRIEN ESKUALDEAK



- Eskualdeen gaineko eragiketak:
  - Eskualdea sortu
    - Inplizituki hasierako mapa sortzerakoan edo eskari esplizituagatik exekuzio-denboran (adibideak: fitxategi bat proiektatzean, exec)
  - Eskualdea askatu
    - Inplizituki prozesua amaitzean edo eskari esplizituagatik exekuzio-denboran (adibideak: fitxategia desproiektatu, exec)
  - Eskualdearen luzera aldatu
    - Inplizituki pilarako edo eskari esplizituagatik (adib. Heap)
  - Eskualdea bikoiztu
    - POSIXeko FORK zerbitzuak egiten duena

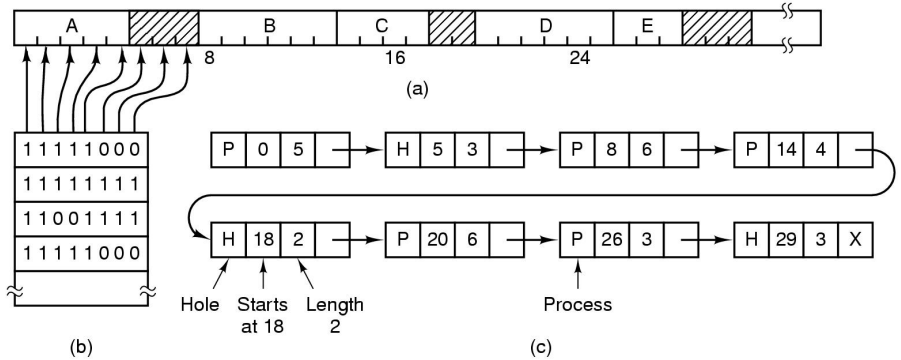
5. MEMORIAREN KUDEAKETA
5.3 ONDOZ ONDOKO MEMORIA-ESLEIPENA

- SE-ak hutsune luze, etenik gabeak, aurkitu behar ditu prozesuei emateko
- Beharrezko hardwarea: Langa erregistroak (Oinarri eta muga erregistroak)
  - Modu pribilegiatuan bakarrik atzigarri
  - Erabiltzaile moduan ezin daitezke gainditu heziak
    - Modu pribilegiatuan bai (atzipen librea)
    - SE-ak ere erabiltzen ditu prozesuen helbide logikoak itzultzeko sistema deietan.



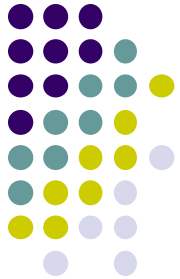
5. MEMORIAREN KUDEAKETA
5.3 ONDOZ ONDOKO MEMORIA-ESLEIPENA

- SE-ak prozesuen memoria kudeatzeko:
  - Langa erregistroen kopiak prozesu bakoitzaren PKB-an gorde
    - Testuinguru aldaketetan SE-ak erregistroetan balio egokiak jarri
  - Memoriaren betetze-egoeraren kontrola
    - Esleitutako guneak eta hutsuneak (a) identifikatzeko datu-egiturak
      - **Bit-mapak** (b) edo **zerrenda estekatuak** (c)
    - Kokapenaren arabera antolatuta, edo neurrien arabera, edo ohikoenak …



  - Ondoz ondoko esleipenak **kanpo zatiketa** sortzen du:
    - Zati libre txikiegiak, memoriaren degradazioa
    - Irtenbidea: memoria trinkotu → birkokapen dinamikoa

5. MEMORIAREN KUDEAKETA
5.3 ONDOZ ONDOKO MEMORIA-ESLEIPENA

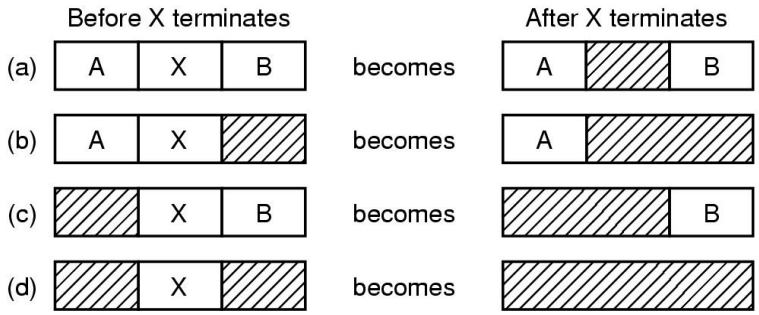


- Memoria esleitzeko politikak:
  - ***First fit***: Nasai sartzeko aurkitzen den lehenengo hutsunea
  - ***Best fit***: Zerrendatik egokia den hutsunerik txikiena hautatu
    - Zerrenda guztia aztertu behar da.
    - Alternatiba: Hutsuneen zerrenda kokapenaren arabera antolatu beharrean, luzeren arabera antolatu (*first fit*-en berdina)
    - Arazoa: Hutsune gauzaezak sortzen dira
  - ***Worst fit***: Libre dagoen hutsunerik handiena esleitu
    - Zerrenda guztia aztertu behar da.
    - Alternatiba: Zerrenda luzeraren arabera antolatu

  - First fit da eraginkorrena esleipen-abiadura aldetik, eta, memoria ondo aprobetxatzen du

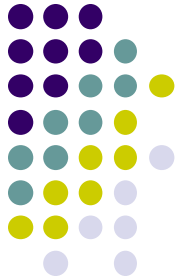
5. MEMORIAREN KUDEAKETA
5.3 ONDOZ ONDOKO MEMORIA-ESLEIPENA

- Memoria askatzea berriz, astunagoa izan daiteke:
  - Auzoko hutsuneak zerrendetan bilatu
  - Elkartu eta birkokatu (luzeraren arabera)



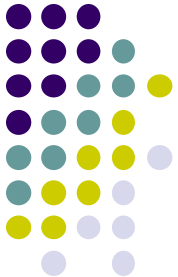
  - Estrategia sofistikatuagoa: ***Buddy*** edo kide sistema
    - Hutsuneak 2-ren potentziako luzerakoak dira.
    - Esleipena: sartzeko gai den 2^k potentziarik txikiena
    - Askapen bizkorra: hutsunearen laguna (luzera berekoa) bilatu eta elkartu
    - Kasu honetan **barne zatiketa** sortzen da: Erabiliko ez den memoria alperrik esleitu.

5. MEMORIAREN KUDEAKETA
5.3 ONDOZ ONDOKO MEMORIA-ESLEIPENA



- Ondoz ondoko esleipenaren balorazioa:
  - Espazio logiko independenteak:
    - Langa erregistroen bidez
  - Babesa:
    - Langa erregistroen bidez
  - Memoria partekatu:
    - Ezinezkoa
  - Eskualdeei euskarri:
    - Guztiak ondoz ondoko kokapenetan
    - Hutsuneetarako ere memoria esleitu behar da
  - Errendimendua maximizatu eta mapa handiak
    - Memoriaren aprobetxamendu txarra, kanpo-zatiketa.
    - Memoria fisiko asko behar da multiprogramazio-maila igotzeko.
    - Prozesua osorik egon behar da memorian, hutsune eta guzti, erabili ala ez.

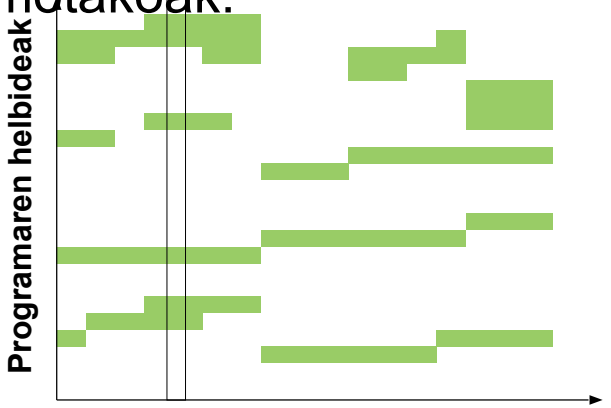
5. MEMORIAREN KUDEAKETA
5.4 SWAPPING ETA MEMORIA BIRTUALA



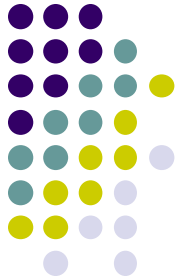
- Zer egin, programa guztien artean, daukagun memoria erabilgarria baino gehiago behar dutenean?
- **Trukea** edo ***swapping***
  - *Swap*: diskoaren partizio bat prozesuen irudiak gordetzeko
  - ***Swap out*** eragiketa:
    - Prozesu aktiboetatik bat swap-era iraultzen da bere irudia osorik kopiatuz (kodea eta hutsuneak ez)
    - Kanporatu beharreko prozesua hautatzeko irizpideak:
      - Prozesuaren lehentasuna
      - Blokeatutako prozesuren bat
      - Prozesuaren maparen gainean DMA eragiketaren bat betetzeke badago ez kanporatu
  - ***Swap in*** eragiketa:
    - Memoria nagusian lekua egiten denean prozesu bat swap-etik ekarri
    - Baita ere prozesu bat luzaroan badago swap-ean:
      - Kasu honetan, beste prozesuren baten *swap out* egin beharko da

5. MEMORIAREN KUDEAKETA
5.4 SWAPPING ETA MEMORIA BIRTUALA

- Zer egin, programa guztien artean, daukagun memoria erabilgarria baino gehiago behar denean?
- **Memoria birtuala**
  - Programaren beharrezko zatiak (blokeak), erreferentziatzen direnak, **memoria nagusian** mantentzen dira eta gainerakoak **memoria sekundarioan** edo diskoan
  - Memoriako bi mailen arteko transferentzia, luzera bereko blokeetan egiten da. Azpitik gora **eskaera** bidez, goitik behera **kanporatzeaz**.
  - Oinarria: prozesuen erreferentziak ez dira uniformeak, **lokalismoa** erakusten dute. Bi motakoak:
    - Lokalismo espaziala
    - Lokalismo denborala



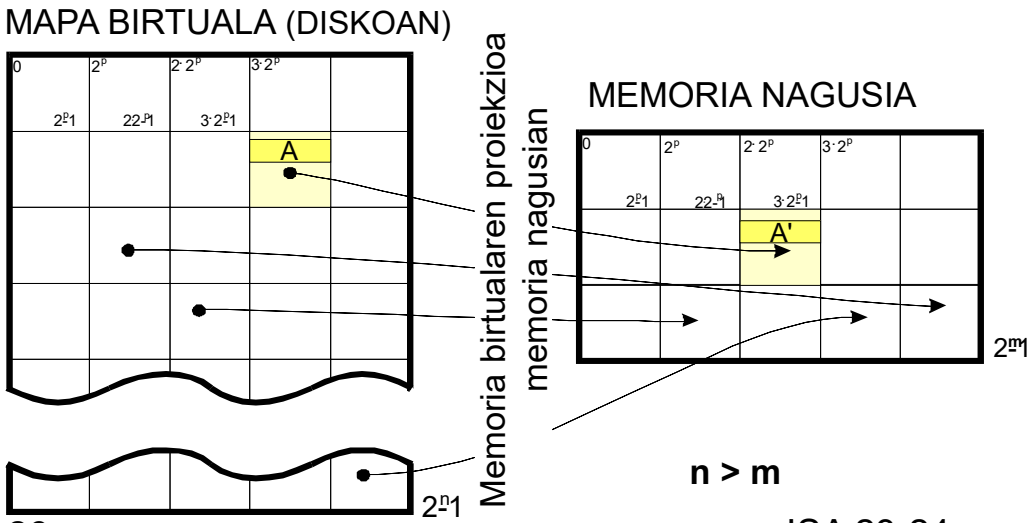
5. MEMORIAREN KUDEAKETA
5.4 SWAPPING ETA MEMORIA BIRTUALA



- Prozesuek epe laburrean bere maparen zati txiki bat baino ez dute erabiltzen
- Erabili behar duten zatia (***lan-multzoa***) memoria nagusian mantentzea da kudeaketa honen asmoa (***multzo egoiliar*** egitea)
- Abantailak:
  - Multiprogramazio-maila hazi egiten da
  - Memoria nagusia baino handiagoak diren programak exekutatu daitezke
- Helbide logiko edo birtual ↔ helbide fisiko astuna.
  - Ez dirudi egokia denbora errealeko sistemetarako.
  - Hardwarea beharrezkoa
- Teknika erabilienak (memoria birtual gabe ere erabil daitezke):
  - Orrikapena
  - Segmentazioa
  - Segmentazio orrikatua.

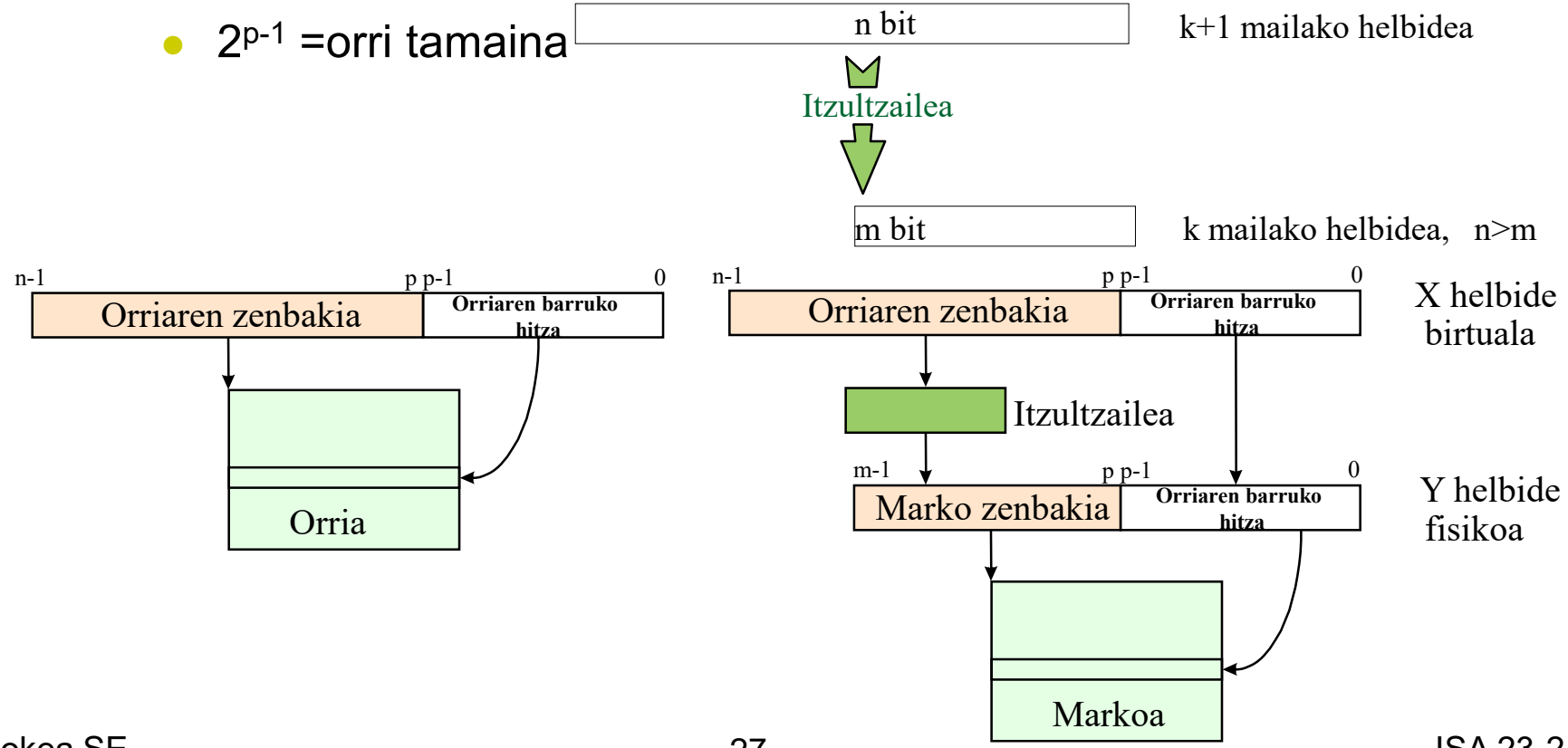
5. MEMORIAREN KUDEAKETA
5.5 ORRIKAPENA

- Memoria espazioa (nagusia+sekundarioa) **orritan** (esleipen-unitatetan) zatitzen da
  - Normalean, ber bi tamainakoak.
- Memoria nagusiko orriak, **markoak** dira
  - MMUk (Memory Management Unit) **helbide birtualak** edo logikoak **helbide fisiko** bihurtzen ditu
  - Helbideratze-espazio birtuala fisikoa baino luzeagoa da
    - Marko-kopurua orri-kopurua baino txikiagoa
    - Markorik gabeko orriak erreferentziatzean MMUk **orri-falta** sortuko du
    - SE-a izango da falta tratatuko duena



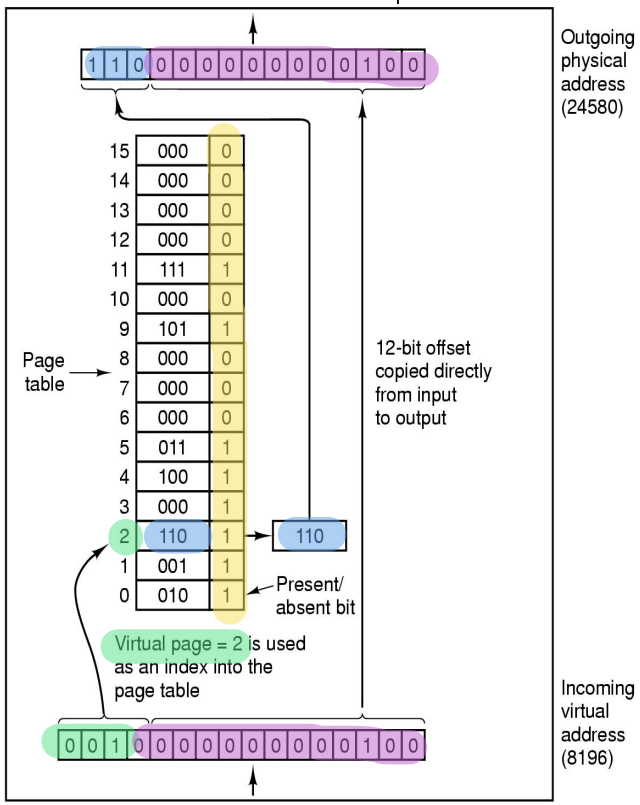
5. MEMORIAREN KUDEAKETA
5.5 ORRIKAPENA

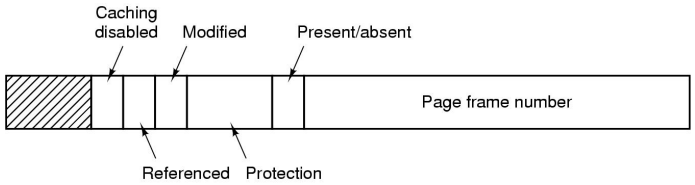
- MMU-k egiten du itzulpena: **orri birtual → orri-marko**
  - Orriaren barruko hitza/byte ez da itzultzen. Eremu horri **offset** deitzen zaio
    - 2^{p-1} =orri tamaina



5. MEMORIAREN KUDEAKETA
5.5 ORRIKAPENA

- **Orri-Taulak** orriak eta markoak lotzen ditu
  - MMU-k orri-taula erabiltzen du itzultzeko
  - Orri-taulako sarreren edukia:
    - Orriarekin lotutako markoaren zenbakia
    - Egote/ez-egote bita
      - 1 → Orriari markoa dagokio
      - 0 → NMI Salbuespena: **orri-falta** → **SE**
    - Babesa: RWX
      - 2 inplementazio aukera:
        - 1 bit (0:R / 1:RW)
        - 3 bit (RWX) CHMOD-ren 3 zenbakiak
      - Baimen ezean → Salbuespena
    - Erreferentziatua izanaren bita (Ref.)
      - MMU-k aktibatzen du erabiliz gero
      - Orri-falta kasuetan erabilgarri
    - Aldatua izanaren bita (Mod.)
      - MMU-k aktibatzen du orrian idatziz gero
    - Cachearen desaktibazio bita:
      - Sarrera hori S/I gailu baten helbidearekin lotuta dagoenean edo DMA eragiketa bat burutzeke dagoenean



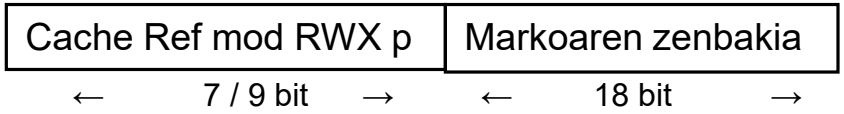


5. MEMORIAREN KUDEAKETA
5.5 ORRIKAPENA – ARIKETA (1/2)

- 32 biteko helbide-busa duen sistema batean 1Gbyte RAM instalatuta dago. Memoria birtuala kudeatzeko 4Kbyteko orriak erabiltzen badira, eta RWX baimen-, mod- eta ref-bitak behar badira, a) lortu helbide logiko eta fisikoen formatua, b) orri-tauletako sarreren formatua, eta c) orri-taularik handienak beharko duen memoria

a) $2^{32}/2^{12}=2^{20}$ orri daude, eta $2^{30}/2^{12}=2^{18}$ marko. Beraz, formatuak:

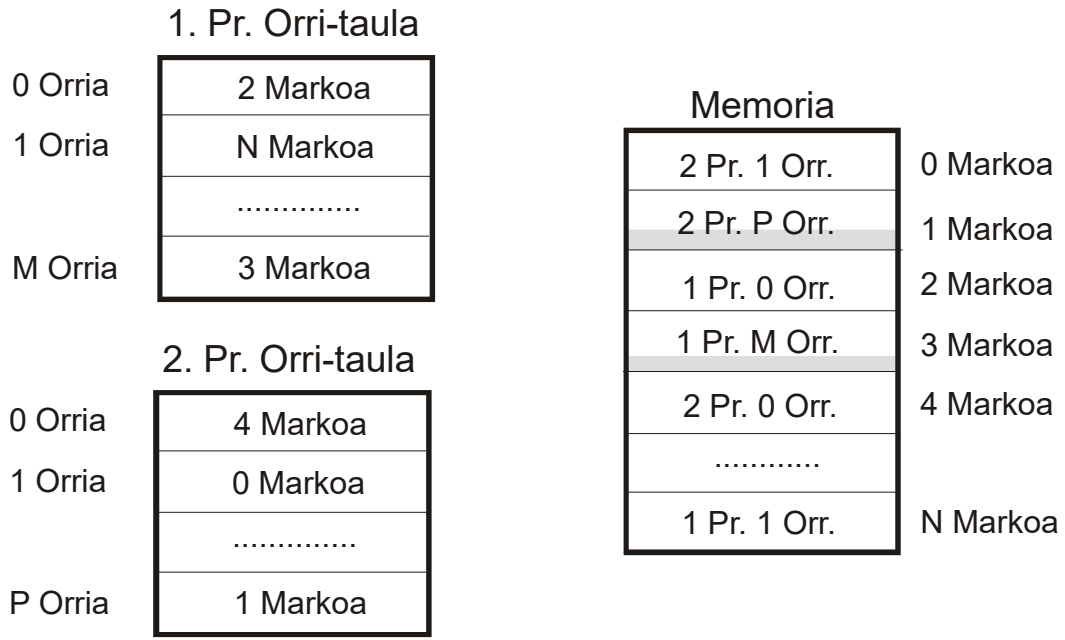
b) **Sarreren formatua**: 18 bit markoa adierazteko + p RWX mod ref cache bitak



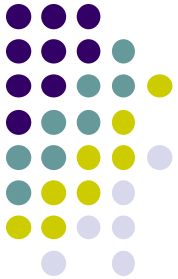
c) 2^{20} orri daudenez, 2^{20} sarrera. Bakoitzak 4byte. Guztira $4*2^{20}$ = 4Mbyte

5. MEMORIAREN KUDEAKETA
5.5 ORRIKAPENA

- Barne zatiketa (txikia) dago:
  - Esleitutako memoria > beharrezkoa
    - Esleitutako markoaren zatiren bat erreferentziatu gabe

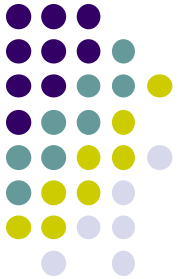


5. MEMORIAREN KUDEAKETA
5.5 ORRIKAPENA



- Orriaren tamaina:
  - Faktore kontrajarriek baldintzatuta:
    - 2^ potentzia eta disko-blokeen multiploa
    - Tamaina txikiaren onurak:
      - Barne zatiketa txikiagoa
      - Lan-multzoa egokiago egiten da egoiliar
    - Tamaina handiaren onurak:
      - Orri-taulak txikiagoak
      - S/I gailuekiko eragiketetan etekin hobea
    - Konpromisoa (2K - 16K)

5. MEMORIAREN KUDEAKETA
5.5 ORRIKAPENA

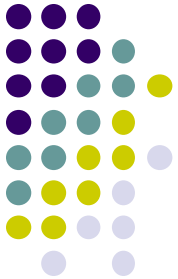


- SE-aren lana:
  - SE-ak prozesuko OT bana mantentzen du
    - Testuinguru-aldaketetan MMU-ari zein OT erabili esan
  - SE-ak OT bakar bat dauka beretzat
    - Nukleo moduan SE-aren OT-z aparte erabiltzailearenak ere atzitu daitezke
  - SE-ak markoen taula mantentzen du:
    - Marko bakoitzaren egoera (libre edo esleituta, ...)
  - SE-ak prozesu bakoitzaren eskualdeen taula mantentzen du
  - Ondoz ondoko esleipenarekin baino gastu handiagoa dauka taulen kudeaketak
    - Funtzionalitate ahaltsuago baten ordaina

5. MEMORIAREN KUDEAKETA
5.5 ORRIKAPENA

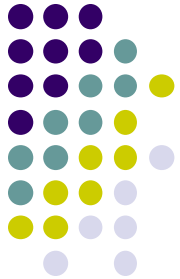
- Orrikapenaren balorazioa:
  - Prozesu bakoitzari memoria-espazio independentea:
    - OT-en bidez
  - Babesa:
    - OT-en bidez
  - Memoria partekatzea:
    - OT desberdinetako sarreretan marko berdinak erreferentziatuz
  - Eskualdeen euskarria:
    - Babes bitak erabiliz
  - Errendimendua maximizatu eta mapa handiak
    - Ez da ondoz ondoko memoria behar
    - Hutsuneek ez dute behar memoriarik
    - Egote/ez-egote bitak
    - Prozesuen haztea
    - Memoria birtualari esker, multiprogramazio maila igo egiten da

5. MEMORIAREN KUDEAKETA
5.5 ORRIKAPENA



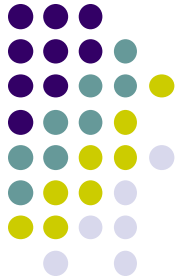
- Orri-taularen inplementazioa:
  - OT-k memoria nagusian mantentzen dira
  - 2 arazo: eraginkortasuna eta biltegiratze-gastua
  - Eraginkortasuna:
    - Atzipen fisiko bakoitzak bi atzipen behar ditu memoria nagusira
      - Orri-taulara + datura edo agindura
    - Irtenbidea: itzulpenak egiteko *cache*a → **TLB** Translation Look-aside Buffer
  - Biltegiratze-gastua:
    - Taulek memoria asko behar dute
      - Adibidean: orriak 4K, hel. logikoak 32bit eta sarrera bakoitzak 4byte:
        - OT-en tamaina: 2^{20} *4 = 4MB/prozesuko
    - Irtenbidea:
      - Maila anitzeko orri-taulak
      - Alderantzizko taula

5. MEMORIAREN KUDEAKETA
5.5 ORRIKAPENA



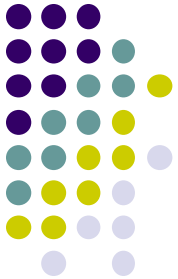
- ***Translation Look-aside Buffer*** (**TLB**):
  - MMU-aren barneko memoria asoziatibo bat
  - Ez da SE-aren ardura
  - Atzitutako azken orrien informazioa gordetzen da bere baitan
    - Atzipenotan erabilitako OT-en sarrerak + orri zenbakia
    - Bi inplementazio aukera:
      - Prozesu identifikatzaileaz:
        - Aurreko informazioaz gain prozesu identifikatzailea ere izango du sarrera bakoitzak
        - SE-ak gehituko du informazio hori
      - Prozesu identifikatzaile gabe
        - TLBa ezabatu beharra dago prozesuz aldatzean (SE)

5. MEMORIAREN KUDEAKETA
5.5 ORRIKAPENA



- ***Translation Look-aside Buffer*** (**TLB**):
  - MMU-ak itzulpena TLB-an aurkitzen badu, atzipen bakarra dago RAMera
    - TLB-a OT-ra kopiatu-> Ref eta Mod balioak
  - Ez badu aurkitzen, MMU-k OT-n bilatuko du
    - Bi atzipen
    - Baliteke kasu honetan TLB-tik sarreraren bat kanporatu behar izatea
  - Aldaera: SW bidez kudeaturiko TLB
    - PUZ batzuek inplementatzen dute
    - MMU-k ez badu itzulpena TLB-an aurkitzen, SE martxan jartzen duen salbuespen bat bidaltzen du
    - MMU-a sinpleagoa da
    - OT-en definizio askea SE-arekin esku

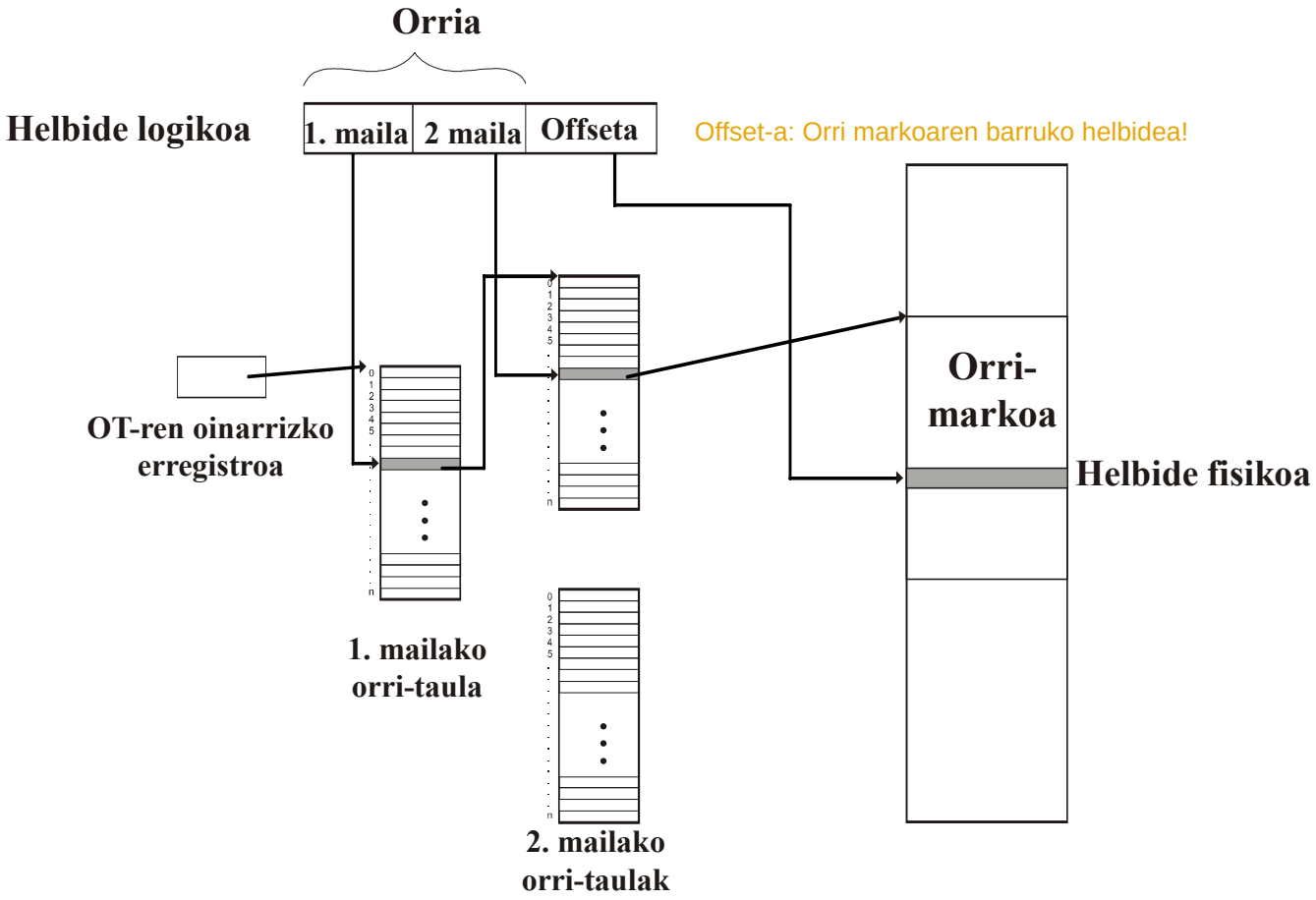
5. MEMORIAREN KUDEAKETA
5.5 ORRIKAPENA



- Maila anitzeko orri-taulak:
  - Ez dago orri-taula bat prozesuko, M mailatan antolatzen dira orri-taulak:
    - K mailako OT-ren sarrerek K+1 mailako OT-k erakusten dituzte
    - Azken mailako taulako sarrerek markoak erakusten dituzte
    - Praktikan, M-ren balioa 2 edo 3
  - Helbide logikoko “orriaren zenbakia” atala, maila bakoitzerako sarrerez osatua egongo da:
    - Maila bakoitzerako eremu bat + offseta
    - Maila bakoitzerako eremuen luzerak ez dute berdinak izan behar
  - Atzipen logiko batek → M+1 atzipen memoriara
    - TLB bidez konponduta
  - Maila bateko OT baten sarrera guztietan ez-egote bita badugu
    - OT hori aurreztu egiten da
    - Goragoko mailako sarreran ez-egote bita jartzen da

5. MEMORIAREN KUDEAKETA
5.5 ORRIKAPENA

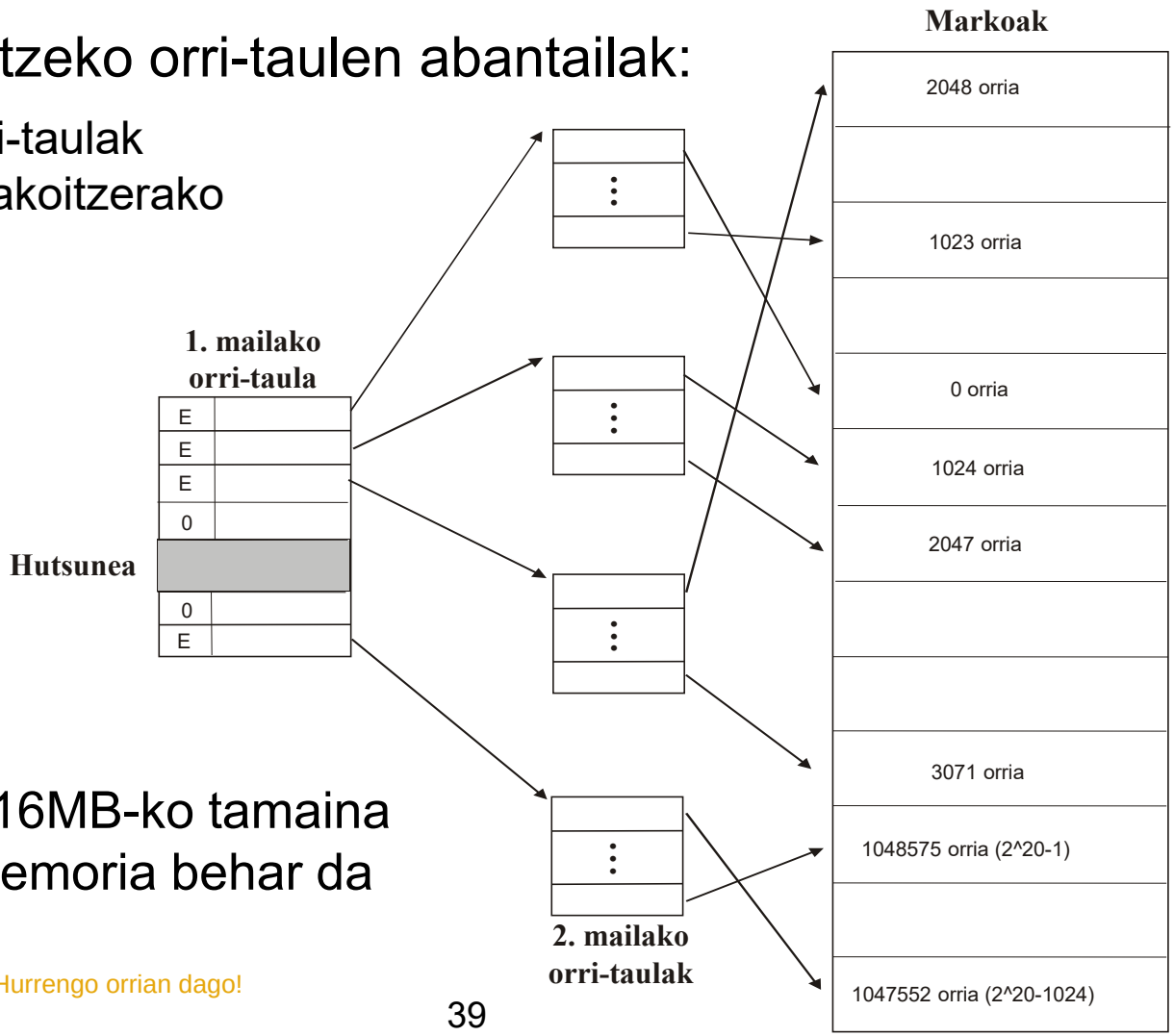
- Maila anitzeko orri-taulak:



5. MEMORIAREN KUDEAKETA
5.5 ORRIKAPENA – ARIKETA (2/2)

- Maila anitzeko orri-taulen abantailak:

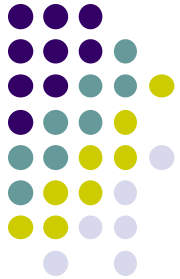
- 2 mailako orri-taulak
- 2^10 maila bakoitzerako



Prozesu batek 16MB-ko tamaina badu, zenbat memoria behar da orri-taulentzat?

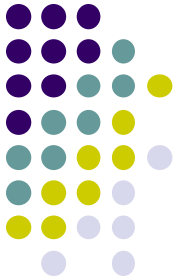
Hurrengo orrian dago!

5. MEMORIAREN KUDEAKETA
5.5 ORRIKAPENA



- Maila anitzeko orri-taulen abantailak:
  - Prozesu batek bere espazio-logikoaren zati txiki bat baino ez badu erabiltzen
    - OT-k gordetzeko behar den memorian asko aurrezten da
  - Ariketa (2/2): Prozesu batek goreneko 12MB eta barreneko 4MB baino ez du erreferentziatzen
    - 2 maila, orriak 4K, hel. logikoak 32bit (10bit mailako) eta sarrera bakoitzak 4byte.
      - Tamaina: 1 1.M OT + 4 2.M OT = 5 * 4KB = 20KB (4MB maila bakarrean)
  - Gainerako abantailak:
    - Memoria partekatua: Tarteko OT-k partekatzea errazten du
    - Bakarrik da derrigorra lehenengo mailako OT memorian edukitzea
      - Gainerako OT-k diskoan egon daitezke eta eskatu ahala memoriara ekarri

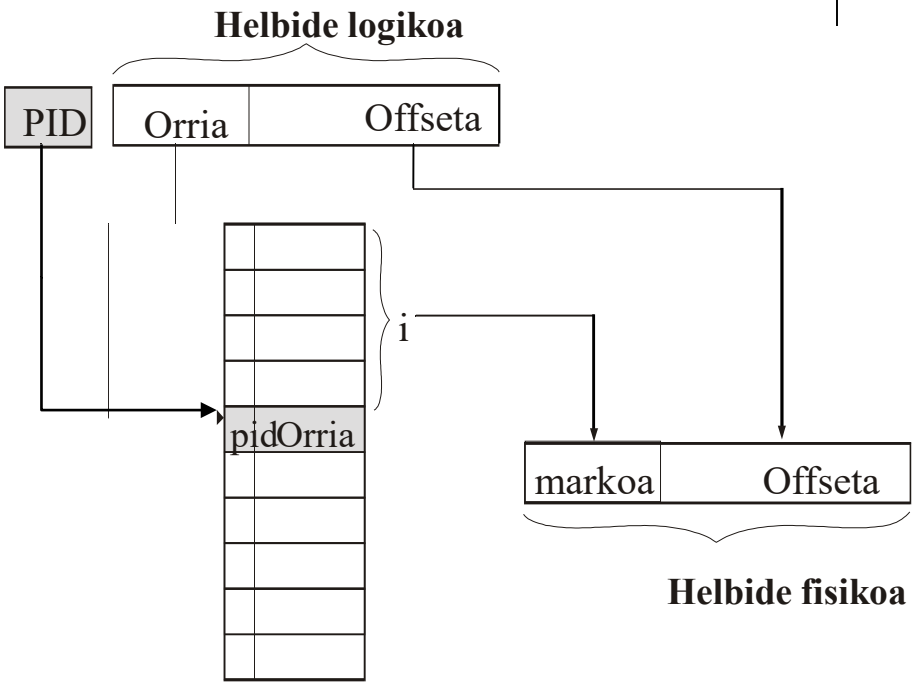
5. MEMORIAREN KUDEAKETA
5.5 ORRIKAPENA



- Alderantzizko taulak:
  - Gaur egungo prozesadoreen helbideratze-espazio logikoa erraldoia da (64biteko helbideak)
    - OT-k, maila anitzeko taulak erabiliz ere, oso handiak.
  - Beste alternatiba bat: Alderantzizko OT-k erabiltzea
    - Taula orokor bakarra, ez bat prozesu bakoitzeko
    - Marko bakoitzeko sarrera bakarra duen taula
    - Sarrera bakoitzaren edukia:
      - Orriaren zenbakia
      - Orri hori lotutako baimenak RWX
      - Dagokion prozesuaren pid-a
    - OT-ren luzera memoria nagusiaren proportzionala da
  - Itzulpenaren prozedura:
    - MMU-k ohiko TLB erabiltzen du
    - TLB-k huts eginez gero → MMU-k alderantzizko OT-n bilatzen du

5. MEMORIAREN KUDEAKETA
5.5 ORRIKAPENA

- Alderantzizko taulak:

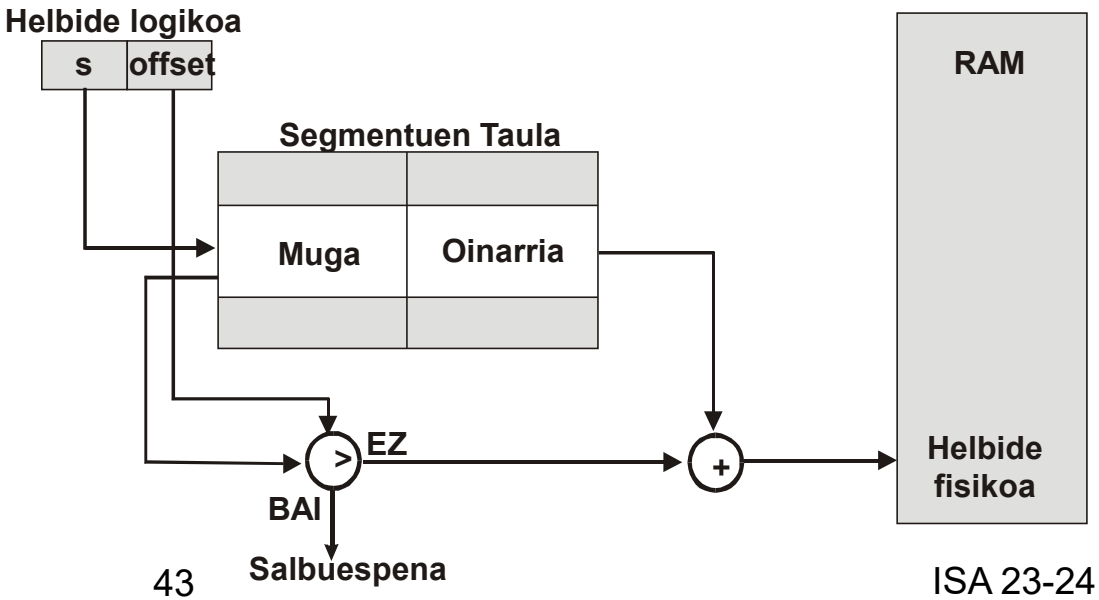


- Desabantailak:
  - Bilaketa sekuentziala behar du
    - *Hash* Taula moduan antolatzen da bilaketa laburtzeko
  - Memoria partekatzea zaildu egiten da
  - Kontuan hartu OT txikiagoa bada ere, SE-k bere orri ez-egoiliarren informazioa nonbaiten gorde beharko duela.

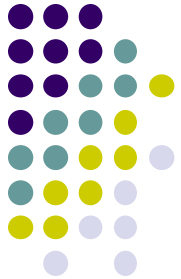
5. MEMORIAREN KUDEAKETA
5.6 SEGMENTAZIOA

Azalpen honetarako, suposatzen dugu segmentazioa eta orrikapena batera EZ ditugula egiten.

- Orrikapenarekin MMU-ak ez ditu ezagutzen prozesuaren eskualdeak, orriak bakarrik ezagutzen ditu
- Segmentazioa eskualdeei euskarri zuzena eskaintzen dien HW teknika bat da
  - Oinarri eta muga erregistroen generalizazioa: segmentuko bikote bana
  - Helbide logikoa: Segmentuaren zenbakia + segmentuaren barruko helbidea



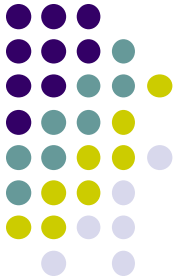
5. MEMORIAREN KUDEAKETA
5.6 SEGMENTAZIOA



- MMU-ak segmentuen taula (ST) bat erabiltzen du. Sarrera bakoitzak (besteak beste) ondoko elementuak ditu:
  - Segmentuaren oinarri eta muga erregistroak
  - Babesa: RWX
- SE-ak prozesuko ST bana dauka
  - Testuinguru aldaketetan zein erabili agintzen dio MMU-ri

- Kanpo-zatiketa: segmentua esleipen-unitatea da (ondoz ondokoa)
- SE-ak memoriaren egoeraren gaineko informazioa gordetzen du:
  - Hutsuneak eta esleitutako guneak identifikatzeko datu-egiturak

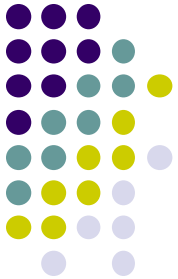
5. MEMORIAREN KUDEAKETA
5.6 SEGMENTAZIOA



- **Segmentazioaren balorazioa:**
  - Prozesu bakoitzari memoria-espazio independentea:
    - Prozesu bakoitzak bere ST
  - Babesa:
    - ST-ak atzipenak mugatzen ditu
  - SEak gainbegiratutako partekatzea:
    - ST desberdinetan sarrera bera errepikaturik
  - Eskualdeei euskarria:
    - Bere punturik sendoena.
  - Errendimendua maximizatu eta mapa handiak: punturik ahulena
    - Kanpo-zatiketa
    - Hori dela eta, praktikan ez da erabiltzen modu hutsean.

5. MEMORIAREN KUDEAKETA
5.7 SEGMENTAZIO ORRIKATUA

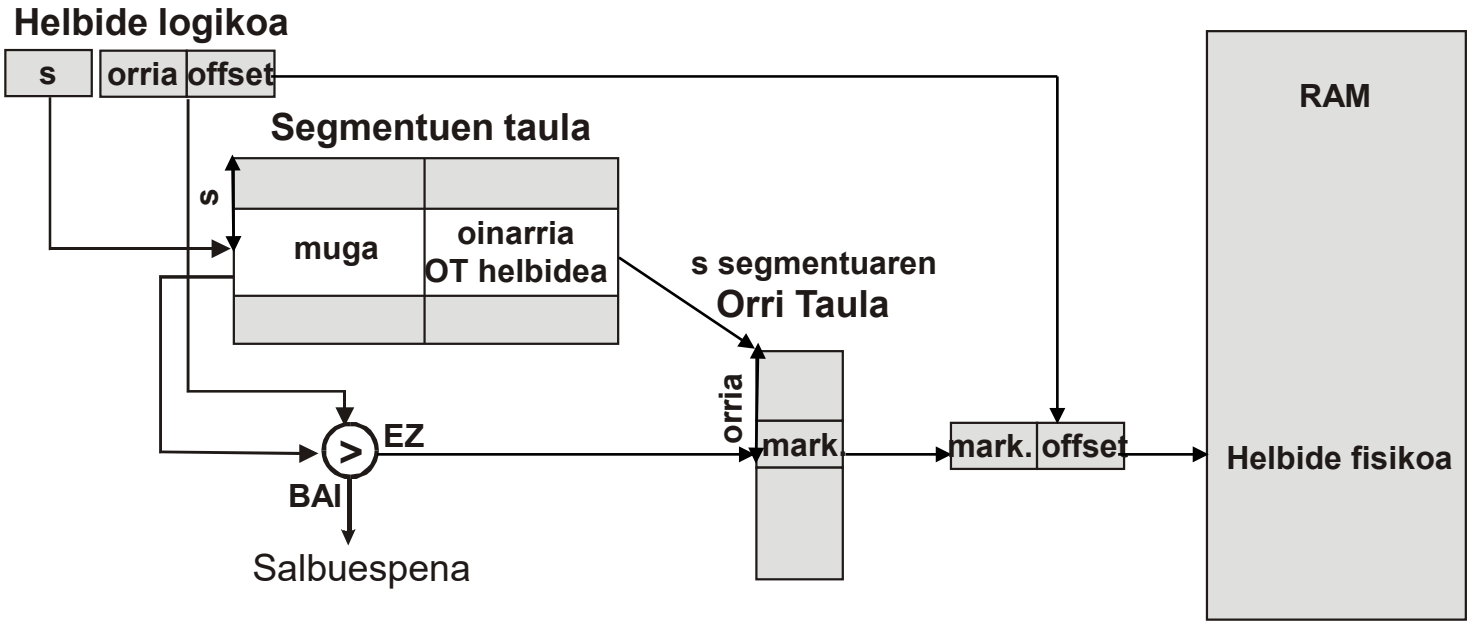
Orokorrean biak batera erabiltzen dira.



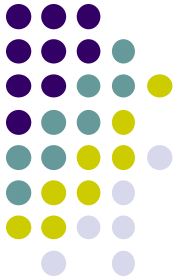
- Aurreko tekniken gauza onak uztartzen ditu:
  - ST-ren sarrera bakoitzak segmentuaren OT erakusten du
  - Segmentazioak: Segmentuei euskarria
    - Eskualdeen gaineko eragiketak erraztu:
      - Babesa: Sarrera bakoitzak bere baimenak
      - Partekatzea: ST biren edo gehiagoren sarrerek OT bera erakusten
  - Orrikapenak:
    - Segmentua ez dago ondoz ondoko kokapenetan
    - Ez dago kanpo-zatiketarik, bai ordea barnekoa (baina txikia)

5. MEMORIAREN KUDEAKETA
5.7 SEGMENTAZIO ORRIKATUA

- Helbide logikoaren eremuak:
  - Segmentua (segmentu-bortxaketak)
  - Segmentuaren barruko orria (orri-faltak)
  - Orriaren barruko offseta

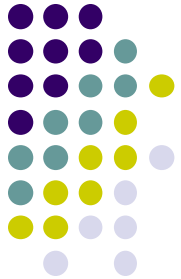


5. MEMORIAREN KUDEAKETA
5.7 SEGMENTAZIO ORRIKATUA



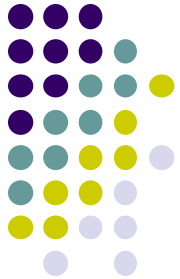
- **Segmentazio orrikatuaren balorazioa:**
  - Prozesu bakoitzari memoria-espazio independentea:
    - Prozesu bakoitzak bere ST
  - Babesa:
    - ST-ak atzipenak mugatzen ditu
  - SEak gainbegiratutako partekatzea:
    - ST desberdinetan OT bera errepikaturik
  - Eskualdeei euskarria:
    - Segmentazioari esker.
  - Errendimendua maximizatu eta mapa handiak:
    - Orrikapenari esker:
      - Ez da egon behar segmentu osoa memorian.
      - Memoria ondo aprobetxatu (barne-zatiketa bakarrik eta txikia)

5. MEMORIAREN KUDEAKETA
5.8 ORRIKAPENA ANTOLATZEKO ALGORITMOAK



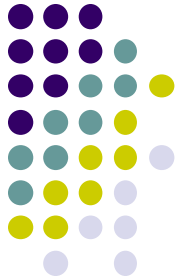
- Memoria birtuala orrikapen edo segmentazio orrikatuaren bidez egiten da (segmentazio hutsa ez)
- Beraz, memoria sekundarioaren eta nagusiaren arteko transferentzia-unitatea orria da.
- Bi **transferentzia-modu**:
  - **Eskaria** edo **demanda**: Egote bita
    - Orri ez egoiliarak Egote bitean 0
    - Atzipenean: Orri-faltako salbuespena
      - SE-ak behar den orria memoria sekundariotik ekartzen du
      - Memoria nagusian esplizituki eskatu diren orriak bakarrik
  - **Aurreorrikapena**: Aldez aurretik ekarri orriak (ez eskari bidez)
    - Orri-falta bat gertatzen denean, aldameneko orriak ekarri
      - Lokalismo espaziala
    - Onuragarria aurresanetan asmatuz gero
      - Gehienetan asmatu egiten da. Asmatu ezik, ez dago kalte handirik

5. MEMORIAREN KUDEAKETA
5.8 ORRIKAPENA ANTOLATZEKO ALGORITMOAK



- Orri-faltaren tratamendua
  - MMU-ak salbuespena sortzen du (helbidea erregistroan)
  - SE-ak etendura horren ardura hartzen du:
    - Helbidea ez bada egokia → Prozesuari SIGILL → akabatu
    - Marko librerik ezean (Markoen Taulan kontsulta)
      - Orriren bat kanporatu beharko da. Marko bati lotuta egongo da.
        - Kanporatu beharreko orrian egote bitean 0 jarri
      - Kanporatu beharreko orria aldatuta badago (*Mod* bitean 1)
        - Memoria sekundarioan idatzi
    - Marko librerik bai (faltaren aurretik edo askatu berria):
      - Orria M markoan kopiatu
      - OT-ko sarreran egote bitean 1 jarri eta M markoan
      - Markoen Taulan M okupatuta dagoela markatu (ez bazegoen)
  - Orri-faltak diskoko bi S/I eragiketa eragin ditzake

5. MEMORIAREN KUDEAKETA
5.8 ORRIKAPENA ANTOLATZEKO ALGORITMOAK

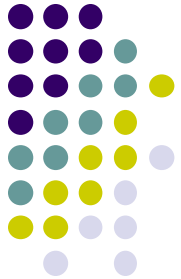


- Memoria birtuala kudeatzeko Politikak

  - **Ordezkapen politika**:
    - Zein orri ordeztu orri-falta batean marko librerik ezean?
    - Ordezpen lokala: bakarrik erabil daitezke prozesuari esleitutako markoak
    - Ordezpen globala: edozein marko erabil daiteke

  - **Prozesuei espazioa esleitzeko politika**:
    - Nola banatu markoak prozesuen artean?
    - Esleipen dinamikoa ala finkoa

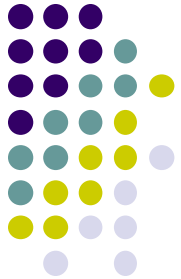
5. MEMORIAREN KUDEAKETA
5.8 ORRIKAPENA ANTOLATZEKO ALGORITMOAK



- **Ordezkapen politikak**:
  - Helburua: Orri-falten kopurua minimizatzea.
  - Algoritmo bakoitzak bertsio lokala eta globala:
    - lokala: irizpidea prozesuaren orriekin bakarrik
    - globala: irizpide bera orri egoiliar guztiekin

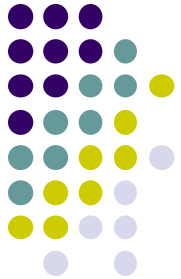
- Algoritmoak
  - Optimoa
  - NRU
  - FIFO
  - Bigarren aukerako FIFO edo Erlojuarena
  - LRU

5. MEMORIAREN KUDEAKETA
5.8 ORRIKAPENA ANTOLATZEKO ALGORITMOAK



- **Ordezkapen optimoa**:
  - Orri-falten kopurua minimizatzeko orri egoiliarren artetik beranduen atzituko litzatekeena kanporatuko da
  - Ezinezkoa
  - Interes bakarra: gainerako metodoen emaitzekin konparaketa analitikoak egitea

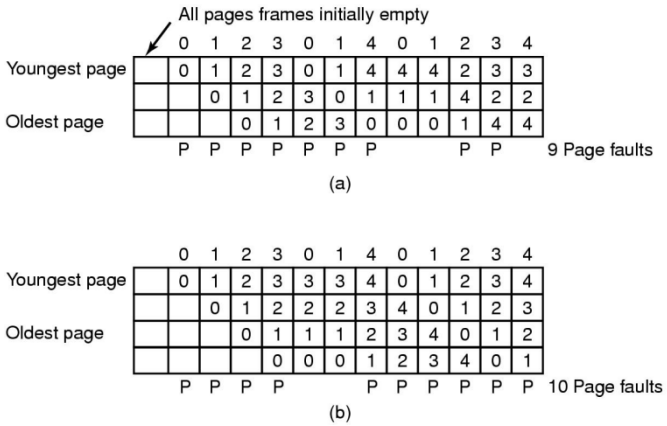
5. MEMORIAREN KUDEAKETA
5.8 ORRIKAPENA ANTOLATZEKO ALGORITMOAK



- **NRU ordezkapena** (*Not Recently Used*):
  - MMU-ak orri bakoitzaren *ref* eta *mod* bitak egokitzen ditu
  - Ondoko orri-sailkapena egiten da:
    1) Ez *ref* Ez *mod*
    2) Ez *ref* Bai *mod*
    3) Bai *ref* Ez *mod*
    4) Bai *ref* Bai *mod*
  - Mailarik baxueneko edozein orri kanporatu
  - *ref* bit guztietan Ez jarri, freskoa izan dadin

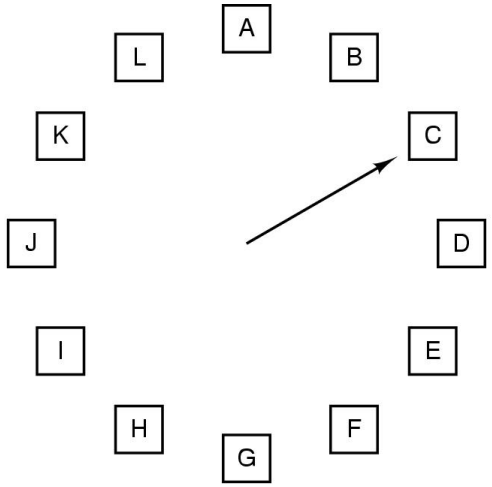
5. MEMORIAREN KUDEAKETA
5.8 ORRIKAPENA ANTOLATZEKO ALGORITMOAK

- **FIFO ordezkapena**:
  - Orri egoiliarren artetik zaharrena kanporatu
  - Inplementatze erraza:
    - Orri egoiliarrak FIFO ordenan → buruan dagoena kanporatu
    - Ez du behar atzipeneko bita (*Ref*)
- Ez da estrategia ona:
  - Zaharrena izan arren, besteak baino maizago erabiltzen bada
  - Irizpidea ez da oinarritzen orriaren erabileran
- Belady-ren akatsa
  - Marko gehiago izanik akats gehiago lortzen dituen adibideak (salbuespenak) aurkitu daitezke



5. MEMORIAREN KUDEAKETA
5.8 ORRIKAPENA ANTOLATZEKO ALGORITMOAK

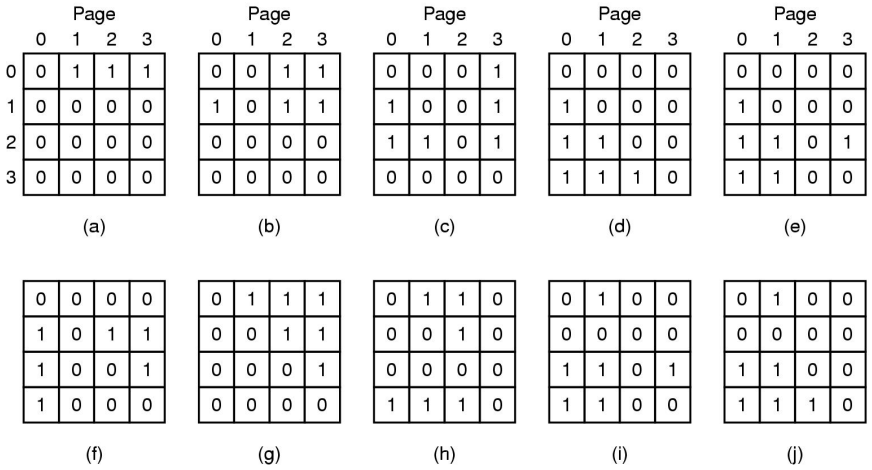
- **Bigarren aukerako FIFO** edo **erlojuarena**:
  - FIFO + *Ref* bita
  - FIFO-k aukeratutako orriaren *Ref* bitean 0 badago
    - Orria kanporatu
  - 1 badago (2. aukera)
    - *Ref* bitean 0 jarri
    - FIFO ilararen amaieran jarri
    - Irizpide bera hurrengo orriari



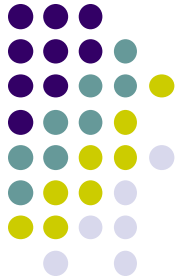
When a page fault occurs, the page the hand is pointing to is inspected. The action taken depends on the R bit:
R = 0: Evict the page
R = 1: Clear R and advance hand

5. MEMORIAREN KUDEAKETA
5.8 ORRIKAPENA ANTOLATZEKO ALGORITMOAK

- **LRU algoritmoa** (*Least Recently Used*):
  - Lokalismo denboralean oinarrituta: orri atzitu berriak berriro atzituak izango dira probabilitate handiz
  - Zaila egitea. HW berezia beharko luke (atzipen bakoitzean gaurkotu behar delako)
  - Software simulazioa (4 markorekin):
    - Atzipen bakoitzean ilara 1z bete eta zutabeak 0z
      - Atzipenak: 0,1,2,3,2,1,0,3,2,3
      - Baliorik txikiena duena kanporatu



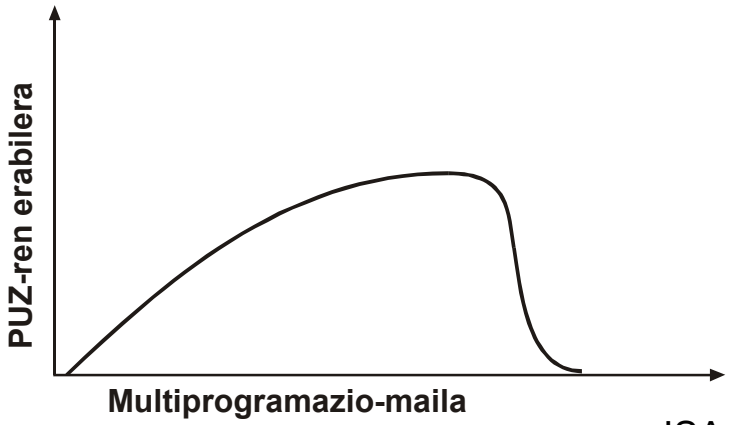
5. MEMORIAREN KUDEAKETA
5.8 ORRIKAPENA ANTOLATZEKO ALGORITMOAK



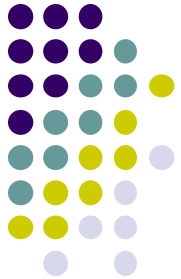
- **Prozesuei markoak esleitzeko politikak**:
  - **Esleipen finkoa**
    - Prozesu bakoitzak marko-kopuru finkoa (multzo egoiliarra)
    - Irizpidea: prozesuaren tamaina, lehentasuna …
    - Ez da moldatzen programaren faseetara
    - Ordezpen lokalarekin bakarrik
    - Arkitekturak multzo egoiliar minimoa eskatzen du:
      - Adibidea: MOVE /DIR1, /DIR2 aginduak gutxienez 3 marko.
  - **Esleipen dinamikoa**
    - Esleitutako marko-kopurua prozesuaren (eta gainerakoen) premietara egokitzen doa
    - Esleipen dinamikoa + ordezpen globala
      - Portaera aurresatea zaila
    - Esleipen dinamikoa + ordezpen lokala
      - Aurresangarriagoa

5. MEMORIAREN KUDEAKETA
5.8 ORRIKAPENA ANTOLATZEKO ALGORITMOAK

- ***Thrashing*** (astindua) edo **hiperorrikapena**:
  - Prozesu batek edo SE-ak orri-falta gehiegi sortzen dutenean
  - Esleipen finkoarekin: P_i -ren hiperorrikapena
    - P_i –ren multzo egoiliarra < P_i –ren lan-multzoa
  - Esleipen aldakorrarekin: Sistemaren Hiperorrikapena
    - Marko-kopurua < Σ P_i –ren lan-multzoa
    - PUZ-aren erabilera jaitsi egiten da. Prozesuak uneoro diskora. Irtenbidea:
      - Multiprogramazio-maila jaitsi eta orri egoiliarrak askatu
  - Nola aurresan egoera hau?



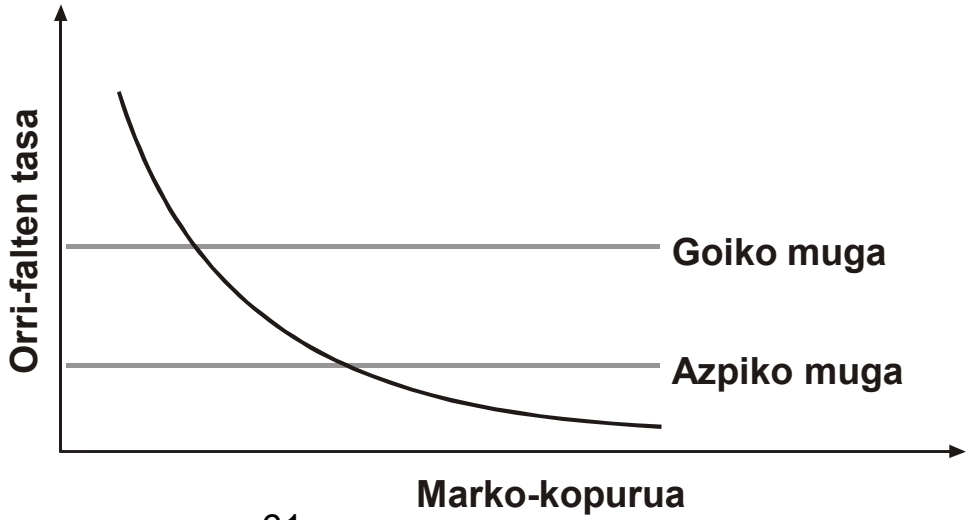
5. MEMORIAREN KUDEAKETA
5.8 ORRIKAPENA ANTOLATZEKO ALGORITMOAK



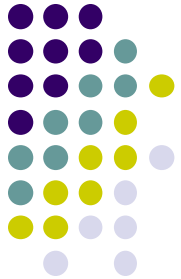
- ***Working Set*** edo Lan-multzoaren estrategia:
  - Prozesuaren lan-multzoa zein den asmatzen ahalegindu prozesuak eginiko azken N atzipenetan oinarrituz
  - N lan-multzoaren denbora-leihoa da
  - Esleipen dinamikoa ordezpen lokalarekin
    - Lan-multzoa txikiagoa egiten bada, markoak askatu
    - Hazi egiten bada berriz, marko berriak esleitu
      - Librerik ez badaude: prozesuren bat bertan behera utzi
      - Markoak berriro askatzen direnean utzitako lanak berreskuratu
  - Inplementazioak MMU berezia beharko luke (LRU-k bezala). SW hurbilketak inplementatu daitezke: PFF.

5. MEMORIAREN KUDEAKETA
5.8 ORRIKAPENA ANTOLATZEKO ALGORITMOAK

- **PFF** (*Page Fault Frequency*) Orri-Falten Maiztasuna :
  - Prozesu bakoitzaren orri-falten tasa kontrolatu
    - Tasa < azpiko muga, ordezkapen algoritmoren bat erabiliz orriak kanporatu
    - Tasa > goiko muga, marko berriak esleitu
      - Marko librerik ezean, prozesuren bat bertan behera utzi aldi baterako
  - Lan-multzoaren egokipena falta bakoitzaren ondoren egiten da, ez atzipen bakoitzean

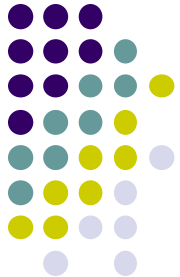


5. MEMORIAREN KUDEAKETA
5.9 MEMORIAN PROIEKTATUKO FITXATEGIAK



- Memoria birtualaren generalizazioa:
  - Programak fitxategia memoria-mapan proiektatzea eskatzen dio SE-ari
    - Babes mota adieraz dezake, baita partekatua ala pribatua
    - Diskoko fitxategia Swap fitxategi bezala erabiltzen da orri-faltak gertatzen direnean
    - Orria marko batean dagoenean atzipena RAM-era da
  - Fitxategiaren atzipena *read*/*write* egin gabe, memorian zuzenean
    - Sistema dei gutxiago behar dira
    - Ez dira behar fitxategi sistemaren cacheko kopiarik
    - Programazioa errazagoa da, datu-egiturak zuzenean RAMean
  - Liburutegi dinamikoen kargan erabiltzen da teknika hau
    - Kodea partekatutako memoria bezala proiektatzen da, X baimenaz
    - Hasierako baliodun datuak berriz, pribatu moduan

5. MEMORIAREN KUDEAKETA
5.9 MEMORIAN PROIEKTATUKO FITXATEGIAK



- Fitxategi bat memorian proiektatu:

*void *mmap(void *direc, size_t lon, int prot, int indic, int desc, off_t desp);*

  - Memoriako zein helbidetan proiektatu den itzultzen du
  - `direc` non proiektatu agindu. Baldin `NULL`, SE-ak aukeratzen du
  - `lon` proiektatu nahi den byte-kopurua
  - `prot` atzipen-mota (RWX)
  - `indic` proiekzio-motaren informazioa (partekatu/pribatu, etab)
  - `desc` proiektatu beharreko fitxategiaren deskribatzailea fd
  - `desp` fitxategiaren zein kokapenetik aurrera proiektatu

- Desproiektatu prozesu baten helbide-espazio zati bat `direc` helbidetik hasita `direc+lon` helbideraino:

*void munmap(void *direc, size_t lon);*

5. MEMORIAREN KUDEAKETA
5.9 MEMORIAN PROIEKTATUKO FITXATEGIAK

- Jatorrizko fitxategi baten kopia memorian proiektatuz:

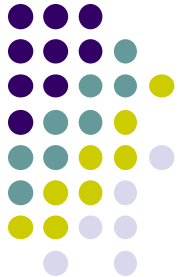
```
void main(int  argc, char **argv)
{
      int i, fdj, fdh;
      char *jatorri, *helburu, *p, *q;
      struct stat bstat;

      if (argc!=3) {
            fprintf (stderr, “Erabilera zuzena: %s jatorritik helburura\n", argv[0]);
            exit(1);
      }

      /* Jatorria ireki irakurtzeko */
      if ((fdj=open(argv[1], O_RDONLY))<0) {
            perror(“Ezin da ireki jatorrizko fitxategia");
            exit(1);
      }

      /* Sortu helburua, 640 baimenaz */
      if ((fdh=open(argv[2], O_CREAT|O_TRUNC|O_RDWR, 0640))<0) {
            perror(“Ezin da sortu helburu fitxategia");
            close(fdj); exit(1);
      }
```

5. MEMORIAREN KUDEAKETA
5.9 MEMORIAN PROIEKTATUKO FITXATEGIAK



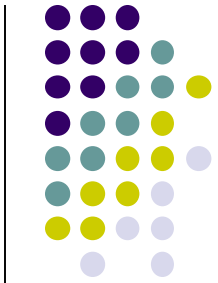
```
if (fstat(fdj, &bstat)<0) {                    /* Jatorrizko fitxategiaren luzera lortu */
        perror("Ezin da irakurri jatorrizko fitxategiaren egoera");
        close(fdj); close(fdh); unlink (argv[2]); exit(1);
}

if (ftruncate(fdh, bstat.st_size)<0) {         /* Helburuaren luzera jatorrizkoaren bera*/
        perror("Errorea luzera finkatzerakoan");
        close(fdj); close(fdh); unlink (argv[2]); exit(1);
}

/* Jatorrizko fitxategia memorian proiektatu */
if ((jatorri=mmap((caddr_t) 0, bstat.st_size, PROT_READ, MAP_SHARED, fdj, 0)) == MAP_FAILED) {
        perror("Errorea jatorriaren proiekzioan");
        close(fdj); close(fdh); unlink (argv[2]); exit(1);
}

/* Helburu fitxategia memorian proiektatu */
if ((helburu=mmap((caddr_t) 0, bstat.st_size, PROT_WRITE, MAP_SHARED, fdh, 0)) == MAP_FAILED) {
        perror("Errorea helburuaren proiekzioan");
        close(fdj); close(fdh); unlink (argv[2]); exit(1);
}
```

5. MEMORIAREN KUDEAKETA
5.9 MEMORIAN PROIEKTATUKO FITXATEGIAK



```
    /* Orain ez dira behar fitxategien deskribatzaileak: itxi  */
    close(fdj);
    close(fdh);

    /* Kopiaren begizta: memoria primarioan hitzez hitz kopiatu, S/I eragiketarik egin gabe*/
    p=jatorri; q=helburu;
    for (i=0; i<bstat.st_size; i++)
            *q++= *p++;

    /* Memoria primarioan burutu den kopia, sekundariora irauli eta proiekzioaren memoria askatu */
    munmap(jatorri, bstat.st_size);
    munmap(helburu, bstat.st_size);
}
```