Computadors
Sistemes Operatius (2/3)

Grau en Ciència i Enginyeria de Dades

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC)

Creative Commons License

Aquest document utilitza Creative Commons Attribution 3.0 Unported License

Els detalls d'aquesta licencia es poden trovar a https://creativecommons.org/licenses/by-nc-nd/4.0

Index

- Components Bàsics del SO
- Gestió de Processos
- **Gestió de la Memòria**
  - **Conceptes Bàsics**
  - Suport del SO
- Gestió de l'E/S i Sistema de Fitxers

Conceptes

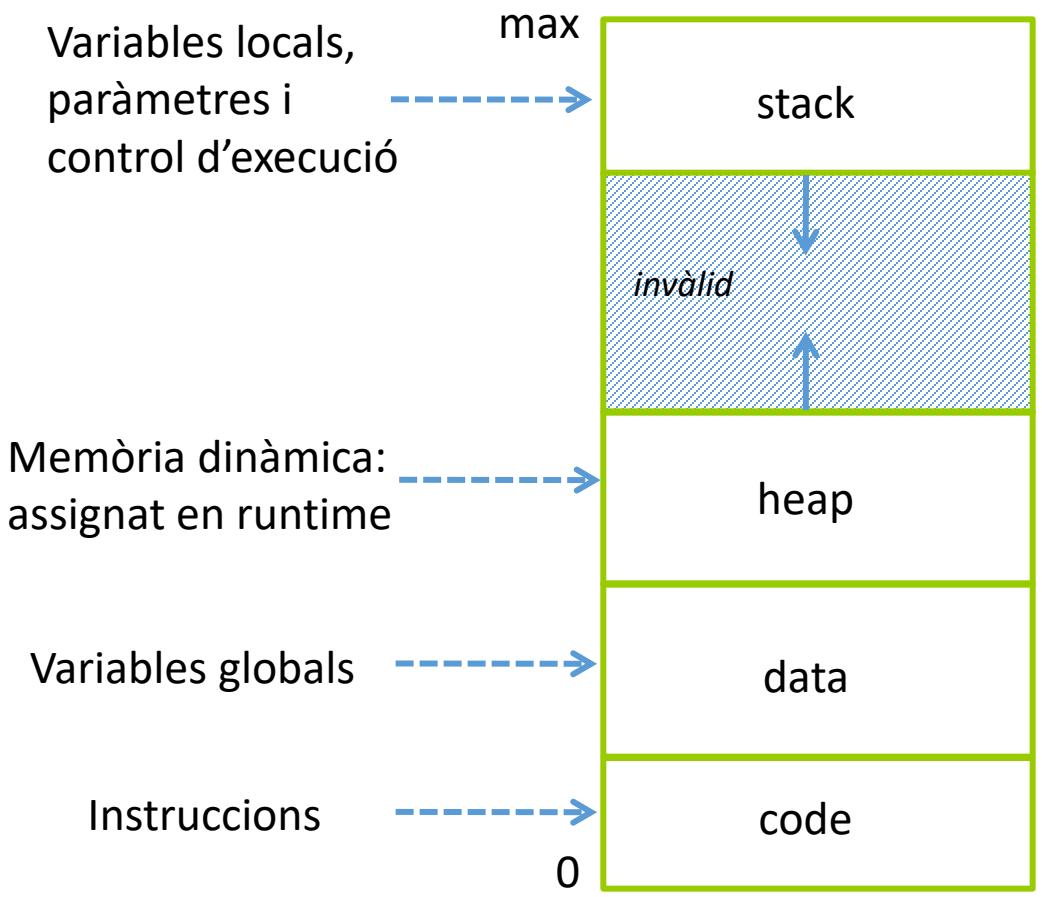
- Espai d'adreces del **Processador**
  - Subconjunt d'adreces que el processador pot emetre (depèn del bus d'adreces)
- El **Procés** és un programa que s'ha llençat a executar
  - El Sistema Operatiu (SO, Sw que s'encarrega de gestionar el sistema informàtic) assigna recursos als processos
    - CPU, **memòria**, dispositius d'E/S, etc.
- Les adreces de memòria emeses per la CPU són **adreces lògiques**
- **Espai d'adreces lògiques del procés**
  - Subconjunt d'adreces lògiques que un procés pot referenciar (el SO quines són vàlides per cada procés)
  - Sw Threads d'un mateix procés comparteixen espai d'adreces lògiques
- Les adreces de memòria que finalment arriben a memòria (RAM) són **adreces físiques**
- **Espai d'adreces físiques del procés**
  - Subconjunt d'adreces físiques associades a l'espai d'adreces lògiques del procés (el SO també és el responsable de decidir quines)

Adreces Lògiques vs Adreces Físiques

- Què passa si són les mateixes (sense diferenciar)?
  - Assignació Fixa: adreces lògiques == adreces físiques

- Què passa si poden ser diferents?
  - Necessita traducció: es pot fer en diferents moments
    - Opció 1, Durant la càrrega del programa:
      - El SO decideix on colocar el procés en memòria i tradueix les referències a memòria durant la càrrega
    - Opció 2, Durant l'execució del programa: cada referència a memòria emesa es tradueix en temps d'execució (runtime). Aquest és el comportament normal en sistemes actuals.
      - Collaboració entre HW i SO
        - HW: proporciona el mecanisme de traducció: **Memory Management Unit (MMU)**
        - SO: ho configura

Espai d'Adreces Lògiques del Procés



Memory Management Unit (MMU)

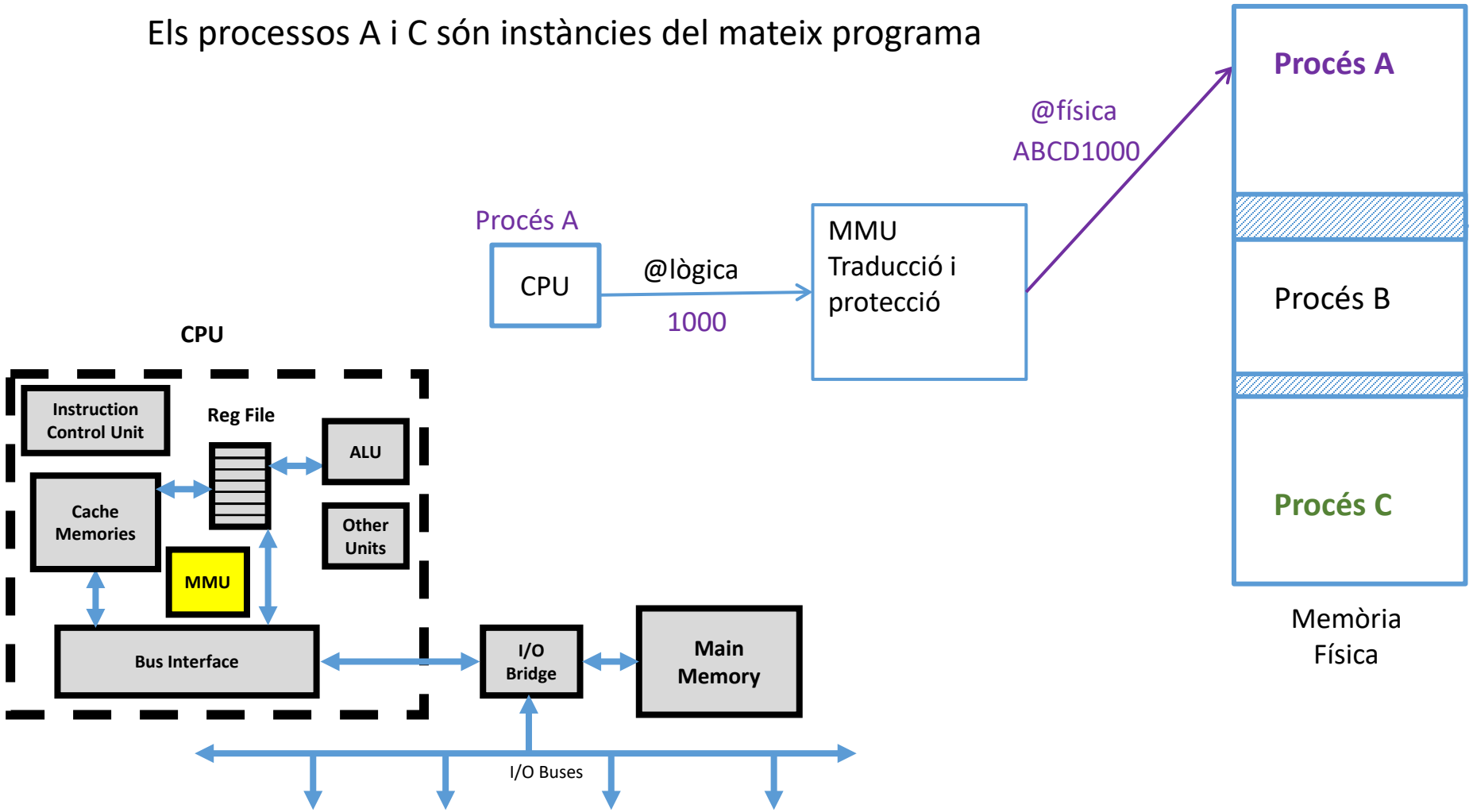
- MMU: És un component HW que, com a mínim, proporciona **traducció d'adreces i protecció als accessos a memòria**.
- **El SO és responsable de configurar la MMU amb els valors de traducció d'adreces corresponents al procés (en concret al software thread) en execució**
  - Quines adreces lògiques són vàlides i quines són les adreces físiques que lis corresponen
  - Garanteix que cada procés se li assigna i accedeix adreces físiques que no són accessibles per cap altre procés
- **Suport del HW per traduir adreces i oferir protecció entre processos**
  - La MMU rep l'adreça lògica i la tradueix a l'adreça física corresponent utilitzant una serie d'estructures de dades
    - Emet un event (també conegut com excepció) al SO si l'adreça lògica no hauria de ser accessible
      - L'adreça està marcada com invàlida, no té els permisos neccessaris per accedir o no està assignada a una adreça física
  - El SO gestiona l'excepció: **Segmentation Fault!!!**
    - **Excepció:** event involutari emès per l'execució d'una instrucció
    - Per exemple, accedir a una adreça de memòria sense permisos corresponents

Sistemes Multiprogramats

- Diferents programes carregats simultàneament en memòria física
- Facilita l'execució concurrent i simplifica el mecanisme de canvi de context
  - 1 procés per hardware thread, però **N processos en memòria física**
  - Quan es canvia el procés que utilitza la CPU (hardware thread) no es necessari tornar a carregar els continguts. Ja estàn en memòria
- El SO ha de garantir **la protecció de la memòria física**
  - Cada procés només pot accedir a la zona de memòria física que té assignada
  - **Col·laboració entre HW i SO**
    - La MMU implementa el mecanisme per detectar accessos il·legals
    - El SO configura la MMU
- El SO ha d'actualitzar la MMU en funció dels canvis que puguin succeir en el sistema, per exemple…
  - … quan es fa un canvi de context (és a dir, canviar el procés que s'executa en un hardware thread), el SO actualitza la MMU per canviar l'informació del procés sortint per l'informació del procés entrant
  - … si l'espai d'adreces d'un procés augmenta

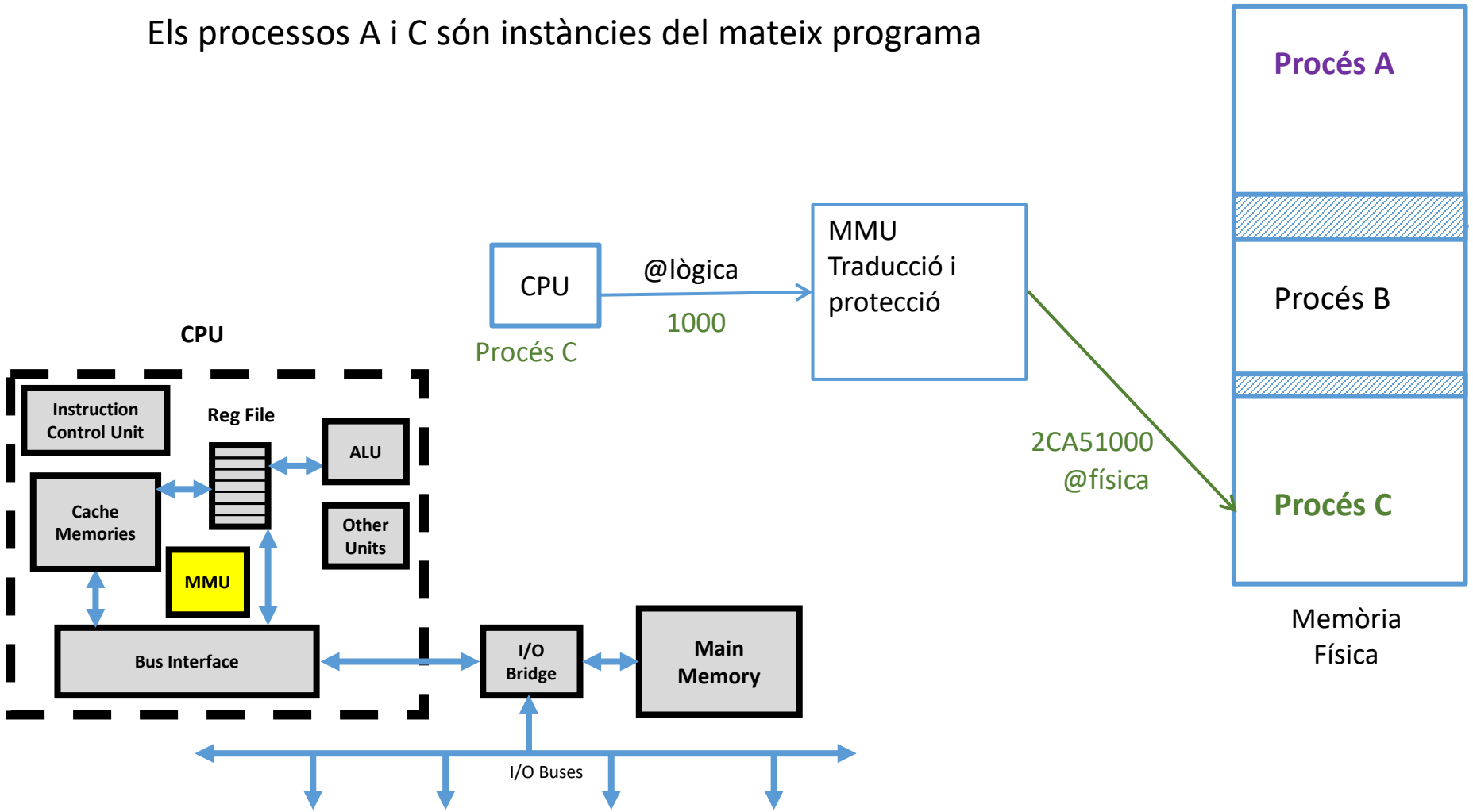
Sistemes Multiprogramats

Els processos A i C són instàncies del mateix programa

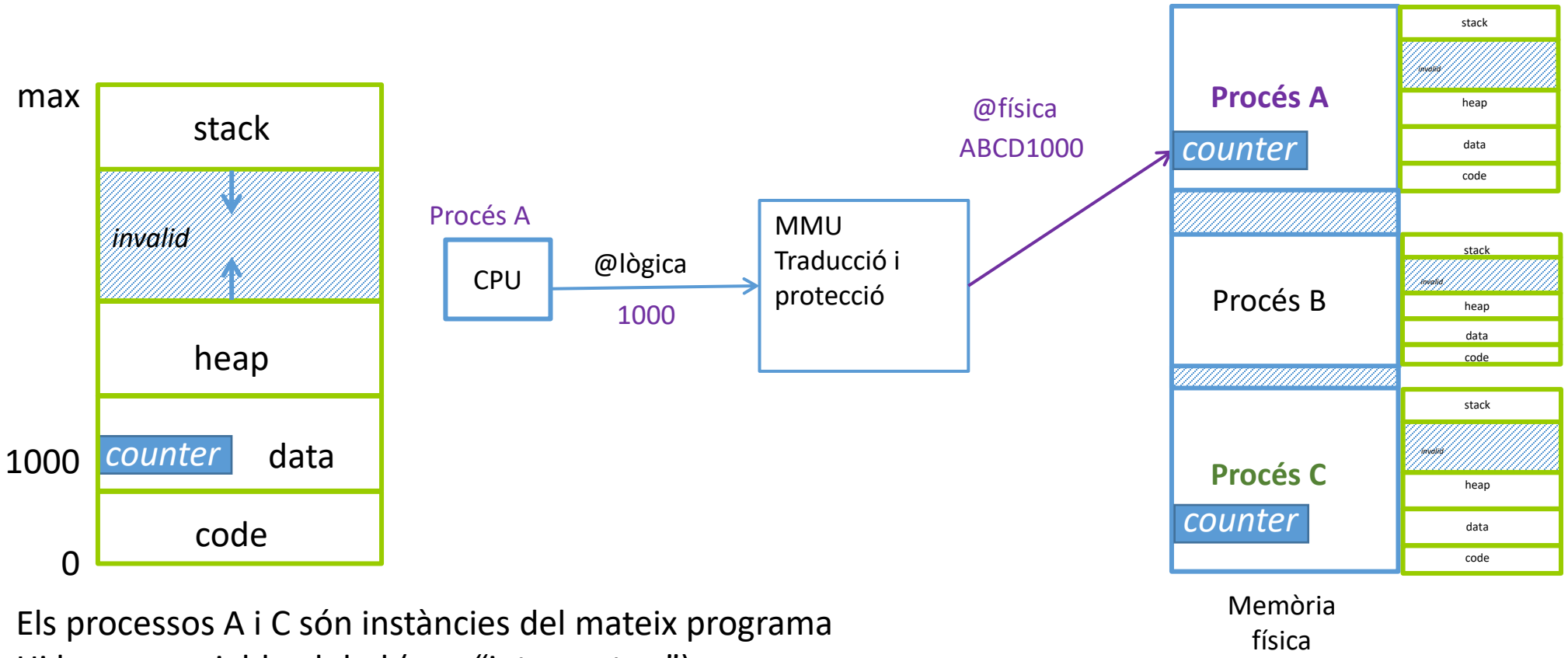


Sistemes Multiprogramats

Els processos A i C són instàncies del mateix programa

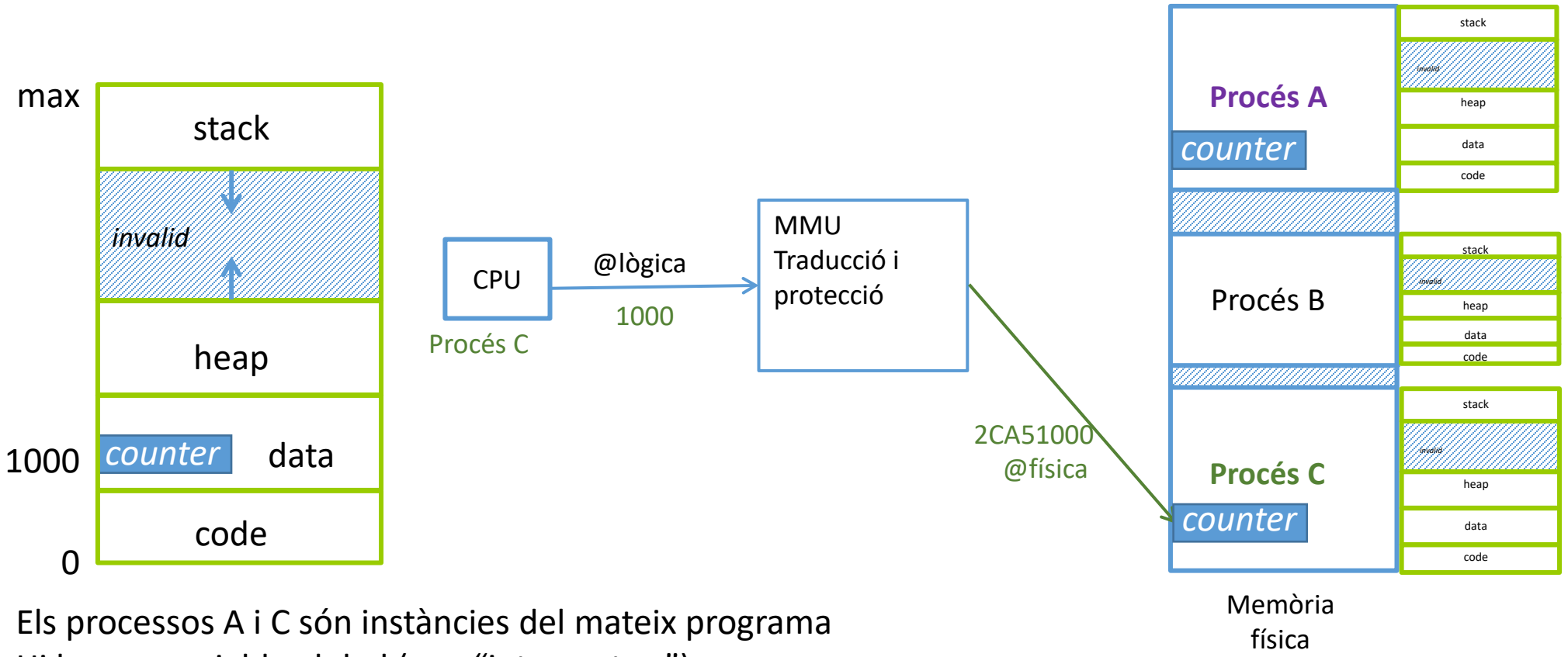


Combinant les coses...



Els processos A i C són instàncies del mateix programa
Hi ha una variable global (e.g. “int counter;”)
- El compilador li ha assignat l’adreça lògica 1000
    - Com és un enter (int), els quatre bytes de la variable van des de @1000 fins a @1003

Combinant les coses...



Els processos A i C són instàncies del mateix programa
Hi ha una variable global (e.g. “int counter;”)
- El compilador li ha assignat l’adreça lògica 1000
    - Com és un enter (int), els quatre bytes de la variable van des de @1000 fins a @1003

Assignació de Memòria

- **Assignació Contigua**
  - L'espai d'adreces físiques és contigu (les adreces són adjacents)
    - Tot el procés es carrega en una zona contigua que es selecciona quan es llença a executar
  - No és flexible i complica aplicar optimitzacions (per exemple, càrrega sota demanda durant l'execució)
- **Assignació No-Contigua**
  - L'espai d'adreces físiques no és contigu (les adreces no necessariament són adjacents)
  - Flexible
  - Impacte en la granularitat de la gestió de memòria dels processos
  - Augmenta la complexitat del SO i la MMU
- Mètodes
  - **Paginació** (particions fixes)
  - **Segmentació** (particions variables)
  - Esquemes combinats
    - Per exemple, segmentació paginada

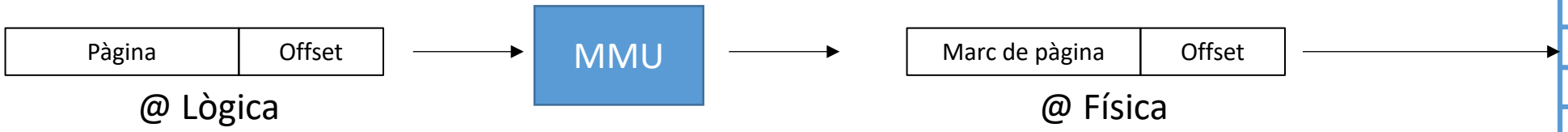
Problemes de l'assignació de memòria: Fragmentació

- **Impacte de la Fragmentació:** quan no és possible satisfer una sol·licitud de memòria encara que hi hagi suficient espai disponible per cobrir-ho. Hi ha memòria lliure, però no pot ser assignada
  - També comentat en la gestió d'emmagatzematge no-volàtil (disc)

- **Fragmentació Interna**: memòria assignada (a un procès) que no s'utilitza

- **Fragmentació Externa**: memòria lliure que no pot ser utilitzada per satisfer la sol·licitud de memòria perquè no hi ha un espai lliure i contigu suficientment gran per la petició feta

Mètode d'assignació: Paginació



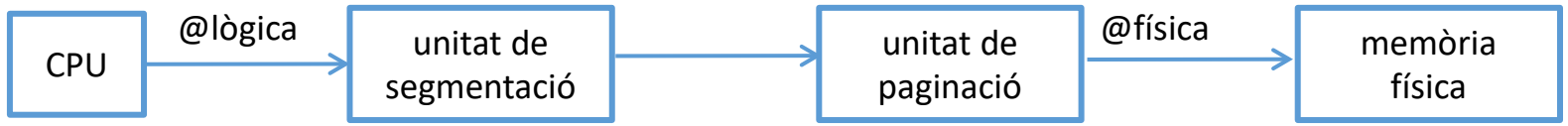
- L'Espai d'adreces lògiques es divideix en particions de mida fixe: **pàgines**
- La memòria física es divideix en particions de la mateix mida: **marcs de pàgina**
- **Pàgina: unitat de treball del SO**
  - Facilita la càrrega sota demanda
  - Habilita la protecció a nivell de pàgina
  - Normalment una pàgina pertanya a una regió de memòria amb els requisits de protecció corresponents (code/data/heap/stack)
- Cada procés té la seva propia **Taula de Pàgines** (es guarden referències bàsiques en el PCB)
  - Es guarda a memòria (RAM) i pot arribar a ocupar MOLT espai (per exemple MBs o inclús GBs)
- Assignació de memòria:
  - Per cada pàgina es busca un marc de pàgina (frame) lliure
  - Els marcs assignats a un procés tornen a la llista de "lliures" quan el procés acaba l'execució
- Pot provocar fragmentació interna

Mètode d'assignació: Segmentació

- L'espai d'adreces lògiques es divideix en regions, en funció del tipus de contingut
  - code, data, etc.
- Les regions són particions **(segments)** de mida variable, que ocupa l'espai que realment necessita
  - Com a minim 3 segments: un per codi, un per stack i un per dades
  - Les referències a memòria es composen per un segment i un offset
- Cada procés té la seva propia **Taula de Segments** (es guarden referències bàsiques en el PCB)
- Assignació de memòria: per cada segment en un procés
  - Buscar una partició lo suficientment gran per cobrir tot el segment
  - Possibles polítiques: first fit, best fit, worst fit
  - Selecciona, de la partició, just la quantitate de memòria que necessita per colocar el segment. La resta de la particiò es guarda en la llista de particions lliures
- Pot provocar fragmentació externa

Mètode d'assignació: Esquema Combinat

- Segmentació paginada: traducció en 2 passos

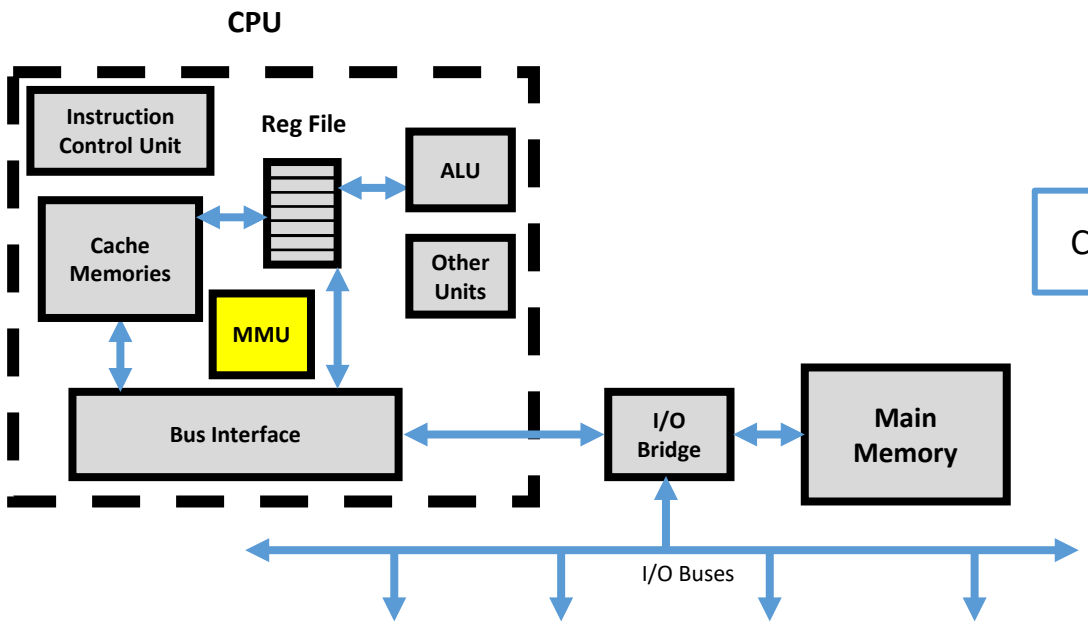


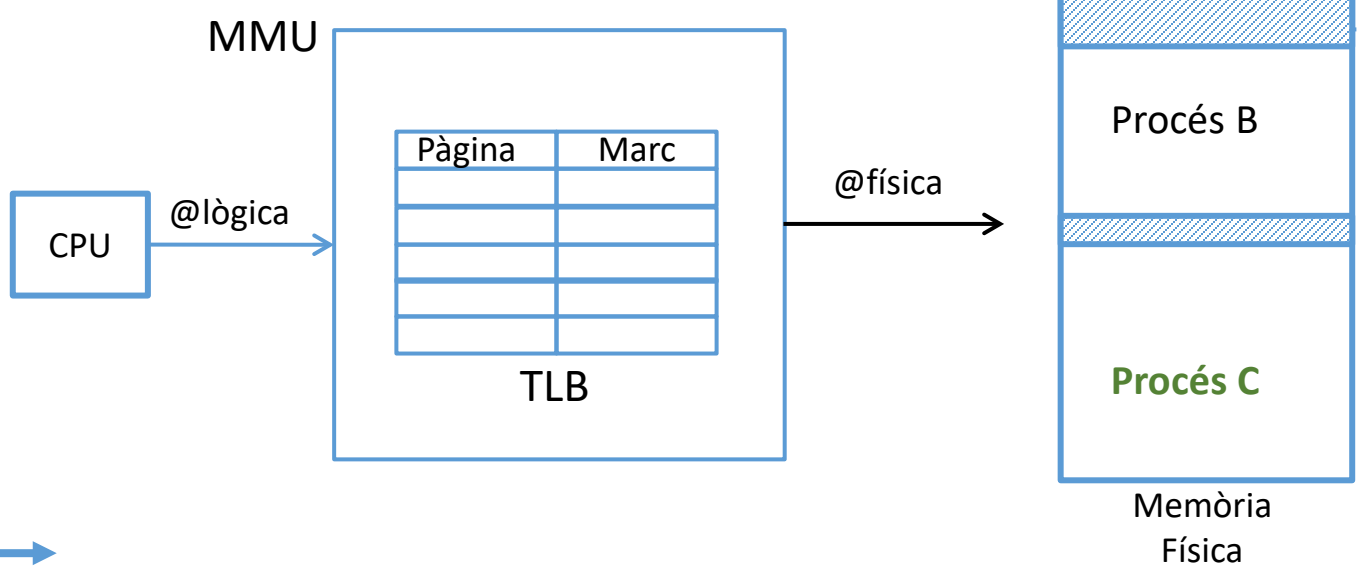
  - L'espai d'adreces lògiques es divideix en segments
    - Code, data, heap, stack
  - Els segments es divideixen en pàgines
    - La mida dels segments és múltiple de la mida de pàgina
    - La página és la unitat de treball del SO

Millores per la gestió d'memòria assignada

La MMU té una "cache" per agilitzar la traducció d'adreces lògiques a físiques

- **TLB: Translation Lookaside Buffer**
  - Memòria cache que guarda les traduccions de pàgina més recents
  - Temps d'accés molt més curt que anar a buscar la Taula de Pàgines (en la RAM)
  - El SO dona suport per la seva gestió (ho veurem més endavant)
    - Especialment en els canvis de context



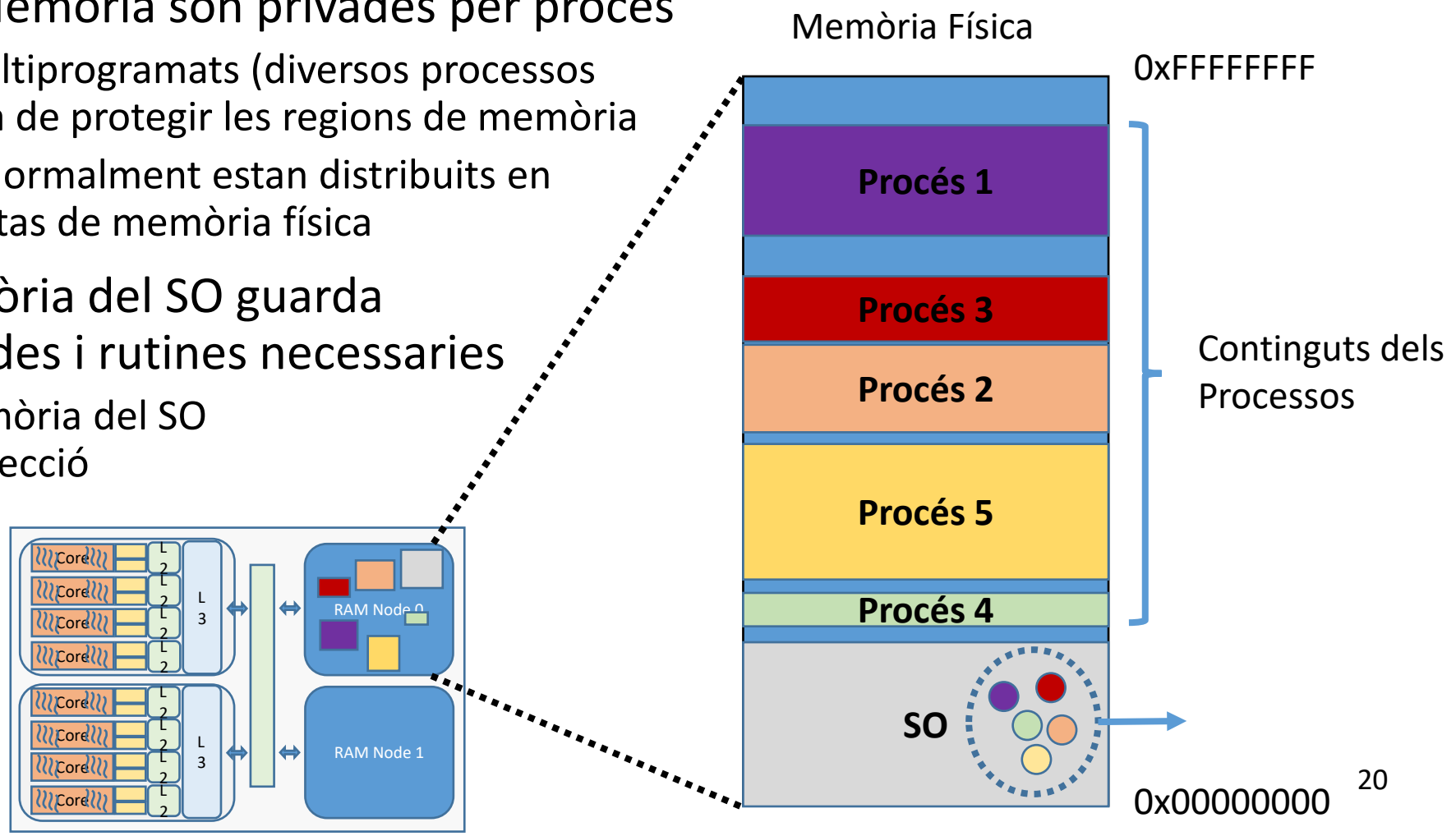


Index

- Components Bàsics del SO
- Gestió de Processos
- **Gestió de la Memòria**
  - Conceptes Bàsics
  - **Suport del SO**
- Gestió de l'E/S i Sistema de Fitxers

Suport del SO

- Les regions de Memòria són privades per procés
  - En sistemes multiprogramats (diversos processos alhora) el SO ha de protegir les regions de memòria
  - Els continguts normalment estan distribuits en posicions remotas de memòria física
- La zona de memòria del SO guarda el kernel i les dades i rutines necessaries
  - La zona de memòria del SO NECESSITA protecció

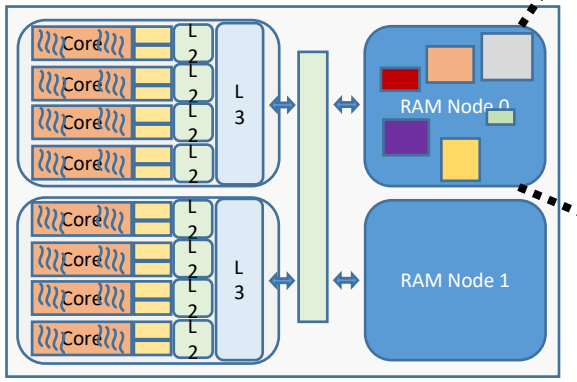


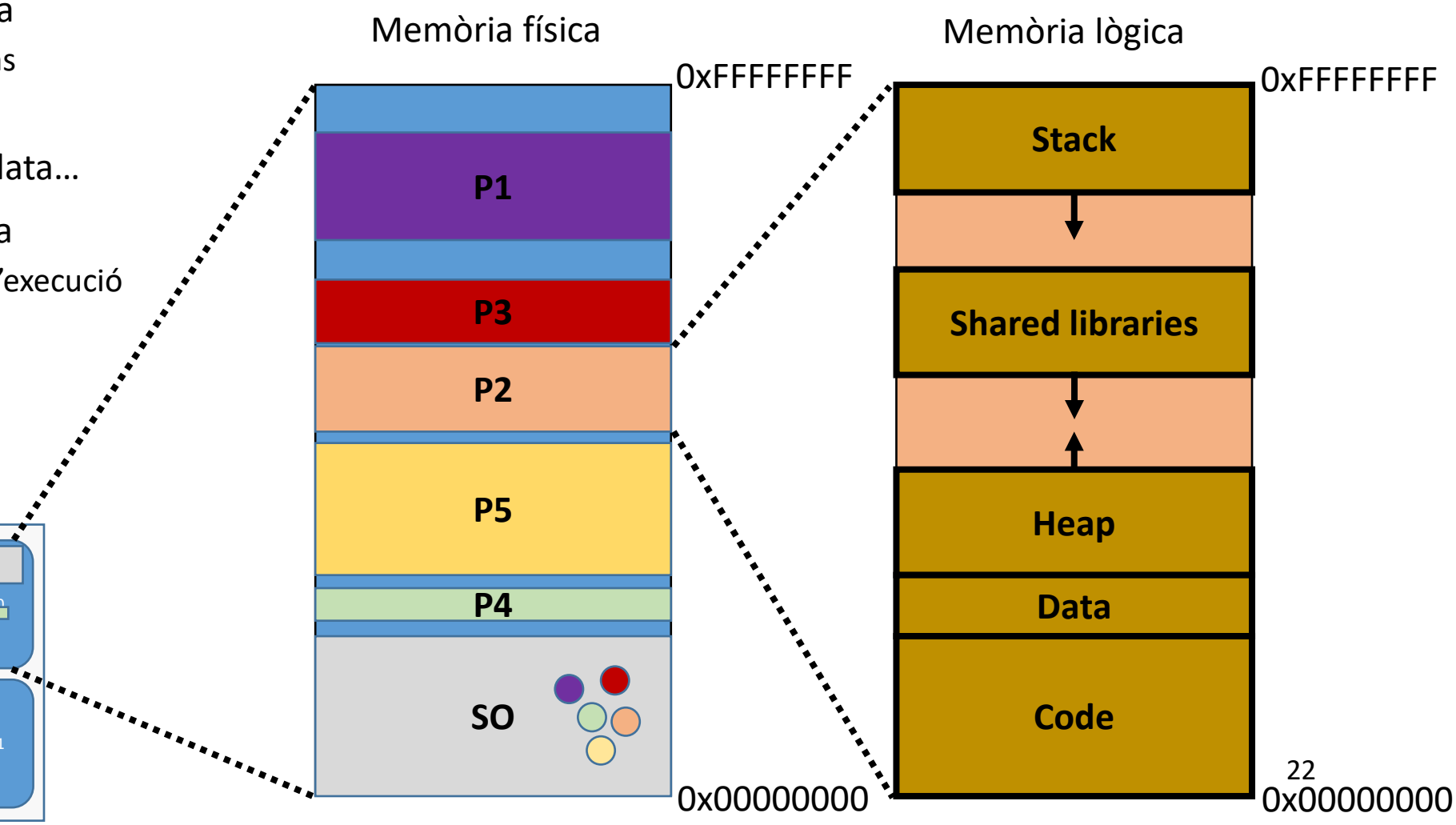
Càrrega d'un nou programa en execució

1. Llegir i interpretar l'executable
2. Crear i inicialitzar el PCB, entre altres, la part de gestió de memòria i assignació de memòria física
3. Inicialitzar la MMU
4. Carregar les seccions del programa des de disc i escriure-les en memòria
5. Inicialitzar els registres (context) del procés (per exemple, el registre program counter)
6. Esperar en l'estat de ready fins que el SO li doni accés a la CPU (canvi de context)

Continguts del Procés en Memòria

- **Stack**: memòria dinàmica
  - Arguments de funcions
  - Variables locals
- **Shared libraries**: Code, data…
- **Heap**: memòria dinàmica
  - Assignada en temps d'execució
- **Data**: .bss & .data
  - Variables globals
- **Code**: .text
  - Instruccions





Syscalls relacionades amb la gestió de memòria

- El Heap s'utilitza normalment per estructures dinàmiques de dades
  - E.g. Estructures només utilitzades per un periode limitat de temps, requisits desconeguts de memòria (quantitat de bytes), assignació i alliberaments frequents de memòria

- Syscalls relacionades amb la gestió del Heap
  - Assignació de Memòria
    - malloc (C library)
    - new (C++ library)
  - Desassignació de Memòria
    - free (C library)
    - delete (C++ library)
  - Totes aquestes crides invoquen una crida al Sistema que modifica el limit del Heap (brk)

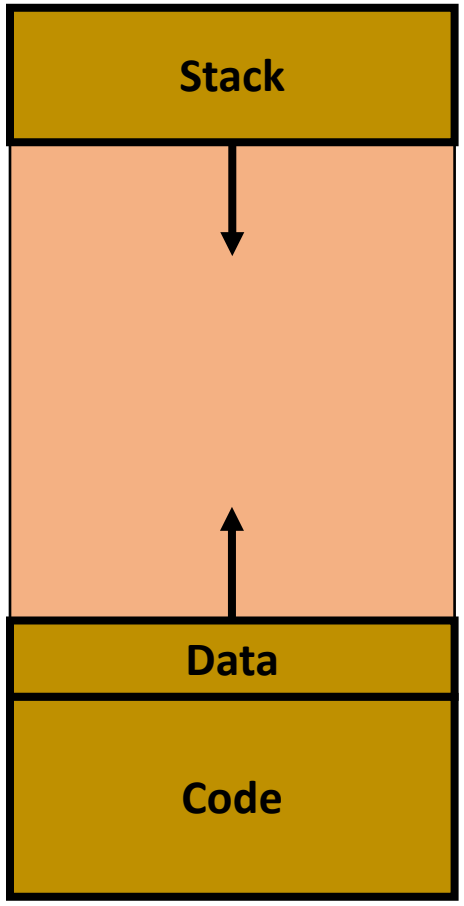
Memory Allocation

- (type) **malloc**(int size); // C language
  - Retorna l'adreça de començament de la zona de *size* Bytes assignats en el Heap
- **new** *type*[size]; // C++ language
  - Retorna l'adreça de començament de la zona de *sizeof(type)*size* Bytes en el Heap

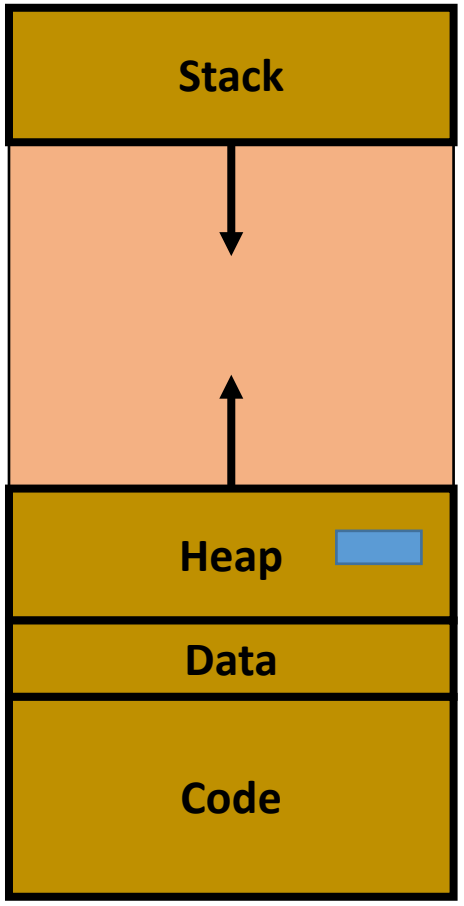
- Funcionament…
  - Abans de l'assignació, comprova si el Heap té espai sufficient per cobrir la sol·licitud
    - En cas contrari, el SO augmenta el limit del Heap
    - La mida del Heap augmenta en una quantitat configurable de Bytes per reduir el nombre de vegades que ho ha de fer. D'aquesta manera es redueix el nombre de syscalls
  - La zona de memòria de l'objecte s'assigna en el Heap a partir d'un algorisme de colocació

Assignació de Memòria

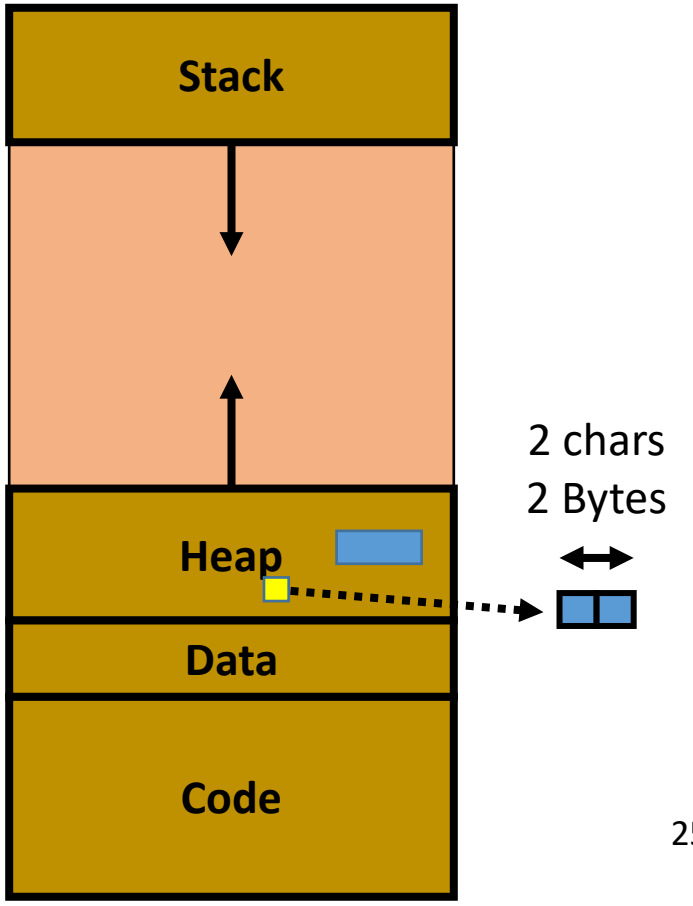
Estat inicial



Després de:
int *ptr = (int) malloc(4*sizeof(int));



Després de:
char *string = (char) malloc(2*sizeof(char));



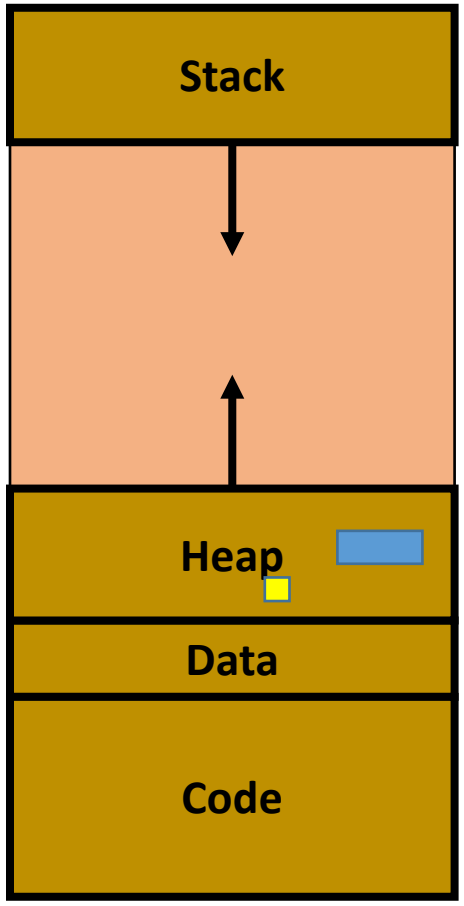
Desassignació de Memòria

- **free**(pointer); // C language
- **delete** [] pointer; // C++ language
  - En els dos casos la zona de memòria apuntada pel punter s'allibera

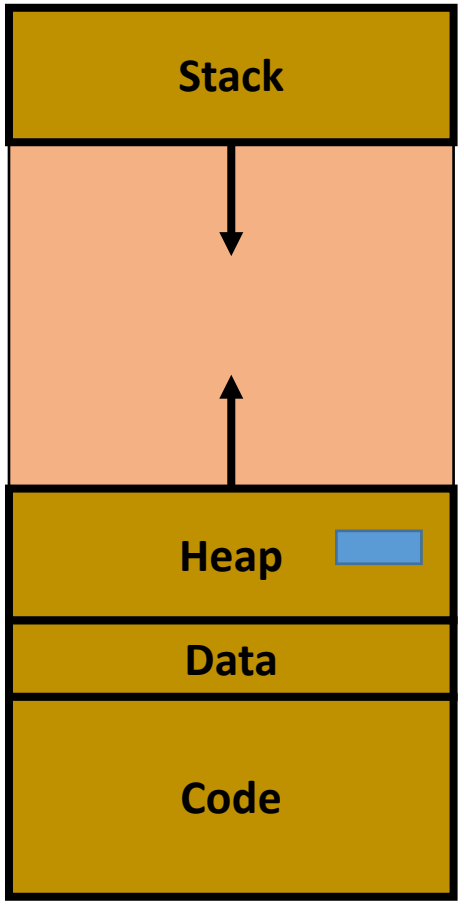
- Funcionament…
  - Desassignació de la zona de memòria apuntada pel parameter d'entrada
  - El SO considerarà reduir o no la mida de la regió de Heap
  - Es mantenen diverses llistes per fer el manteniment de la regió de Heap amb les crides malloc/new/free/delete
    - Llista d'objectes assignats, llista d'objectes desassignats (poden ser ajuntats), espai lliure, etc

Alliberament de Memòria

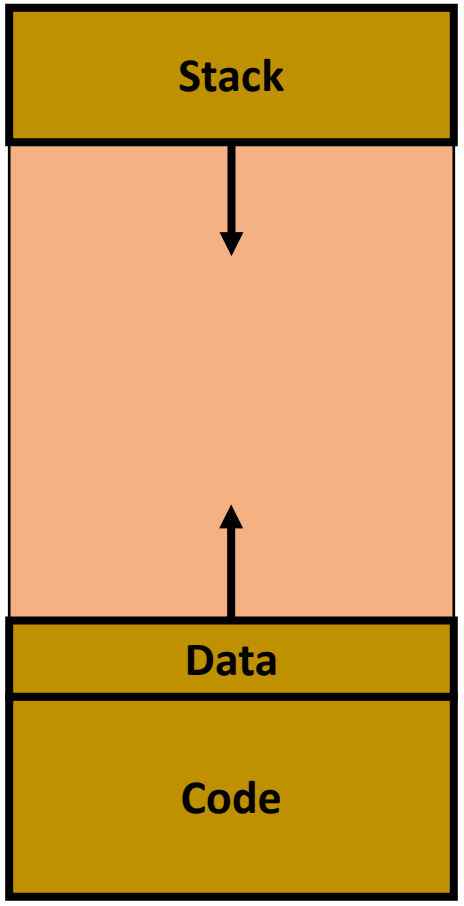
Estat inicial



Després de:
free(string);

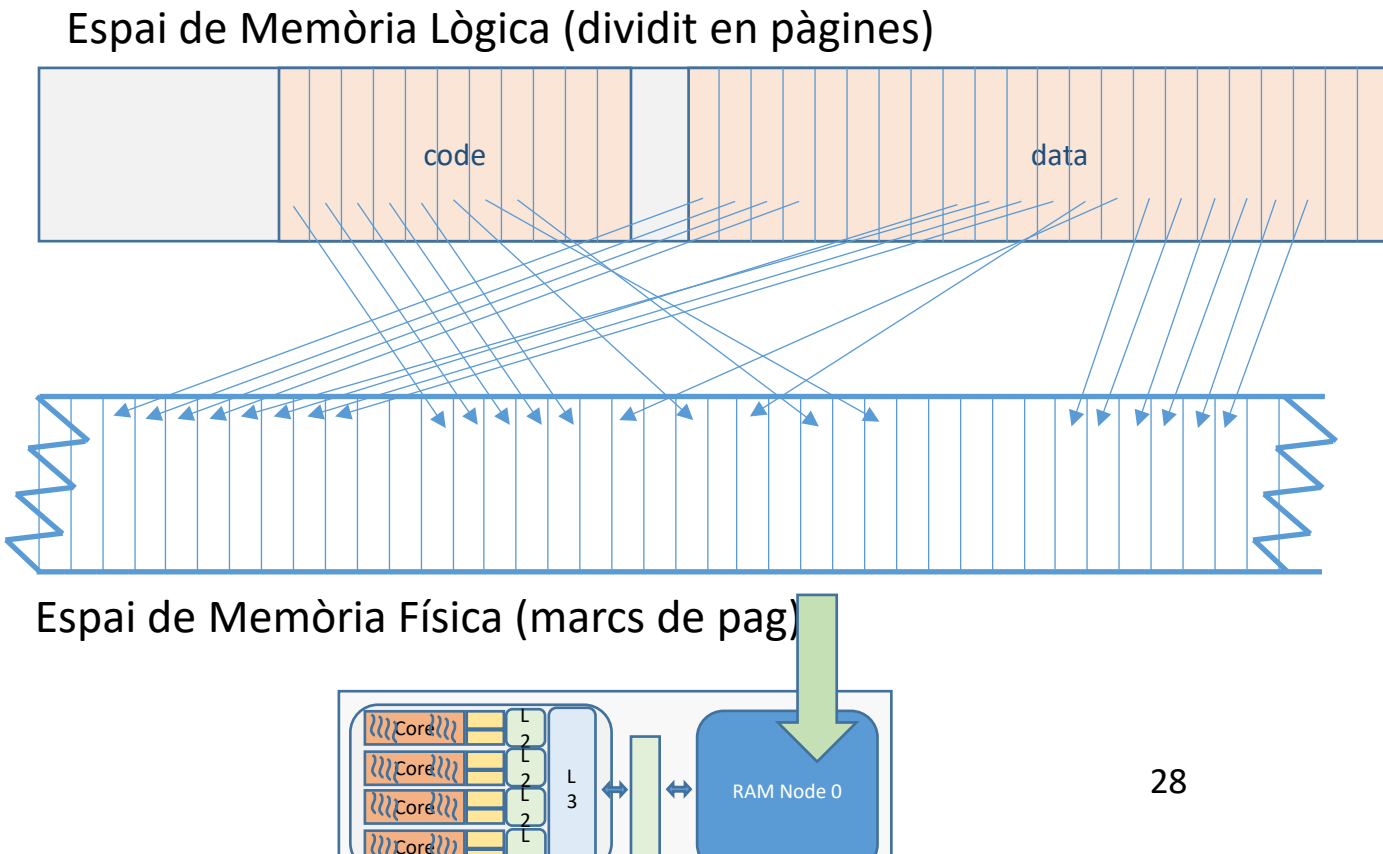


Després de:
free(ptr);



Relació final amb la memòria física

- Memòria Lògica (també coneguda com Memòria Virtual en entorns de SO)
  - Dividit en pàgines (normalment de 4KB, però poden ser de 2-4 MBytes)
  - Una pàgina pot ser...
    - Vàlida i present en la memòria
    - Vàlida i no present en la memòria
    - No vàlida
- Memòria Física
  - Dividit en marcs de pàgina
    - Mateixa mida que les pag.virtuals
  - El SO guarda informació sobre
    - Marcs disponibles
    - Marcs ocupats



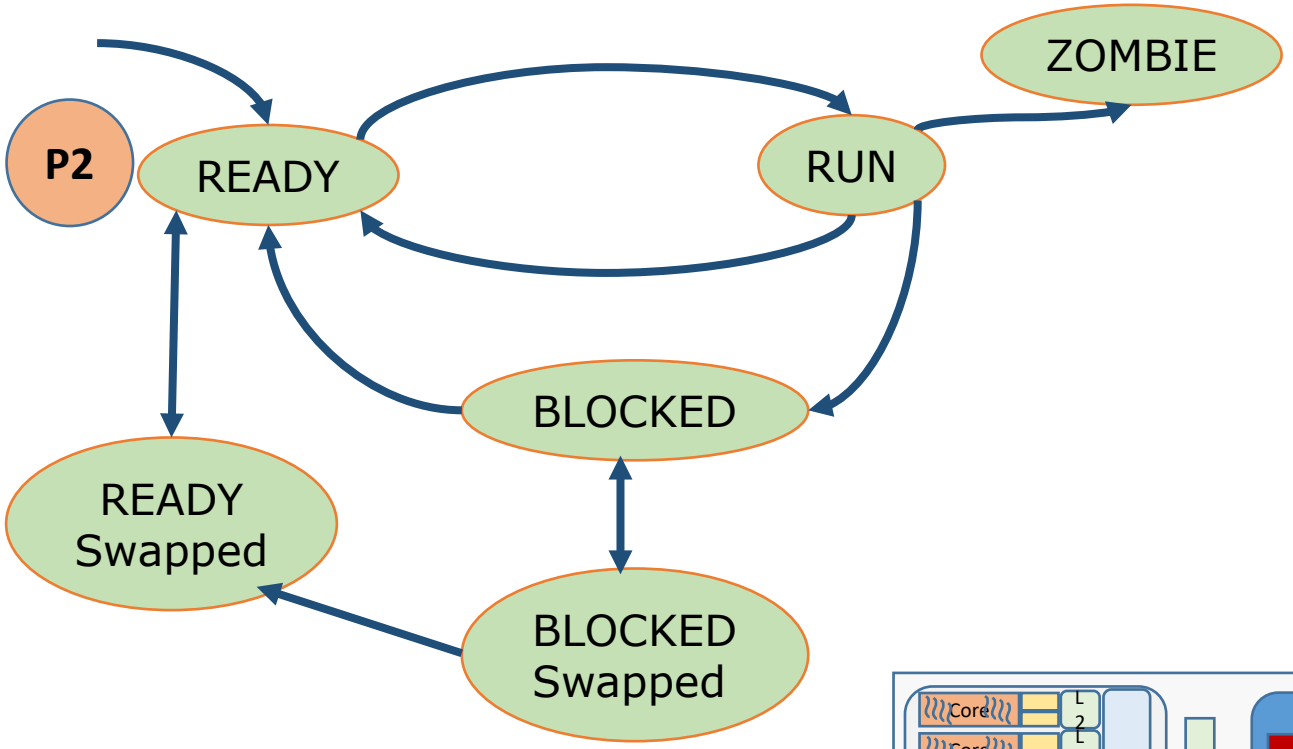
Què passa quan la memòria s'esgota...

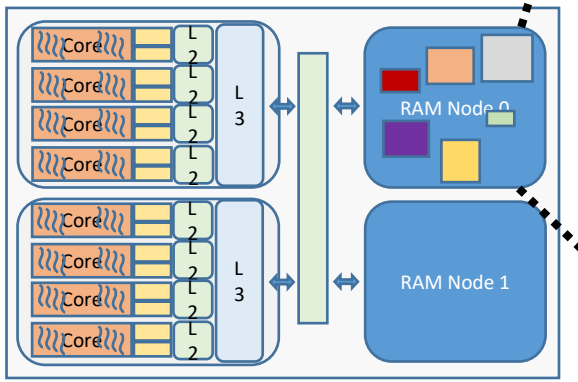
- El SO està constantment comprovant la disponibilitat de memòria del sistema
  - Utilitzat vs disponible

- Si la memòria arriba a esgotar-se (hi ha menys que un determinat Umbral de disponibilitat) el SO comença un procés de gestió alternatiu:
  - Reemplaçar pàgines: paging
  - Reemplagar processos: swapping
  - Es seleccionen pàgines → s'escriuen en **l'area swap** [si es que s'han modificat] → Reassignen altres pàgines

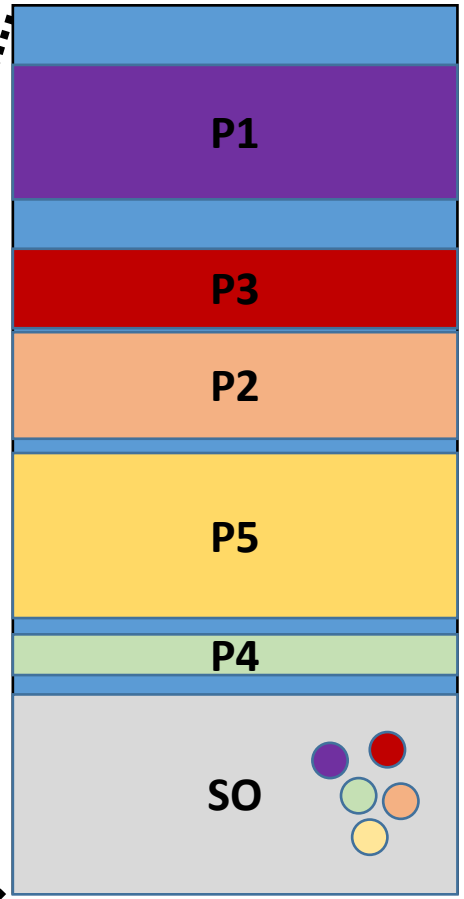
- **Àrea Swap**: àrea de memòria especial, normalment en dispositius de storage (e.g. disc) per guardar temporalment continguts de la memòria principal que no té espai suficient
  - Aquest zona es considera espai addicional a la memòria física
  - Els processos que estàn en àrea swap degut a swapping **NO es poden executar directament**

Exemple de swapping

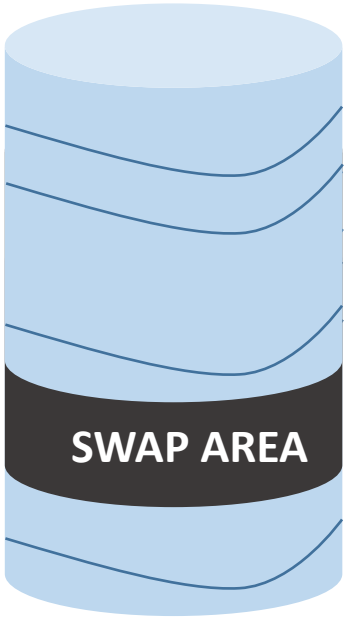




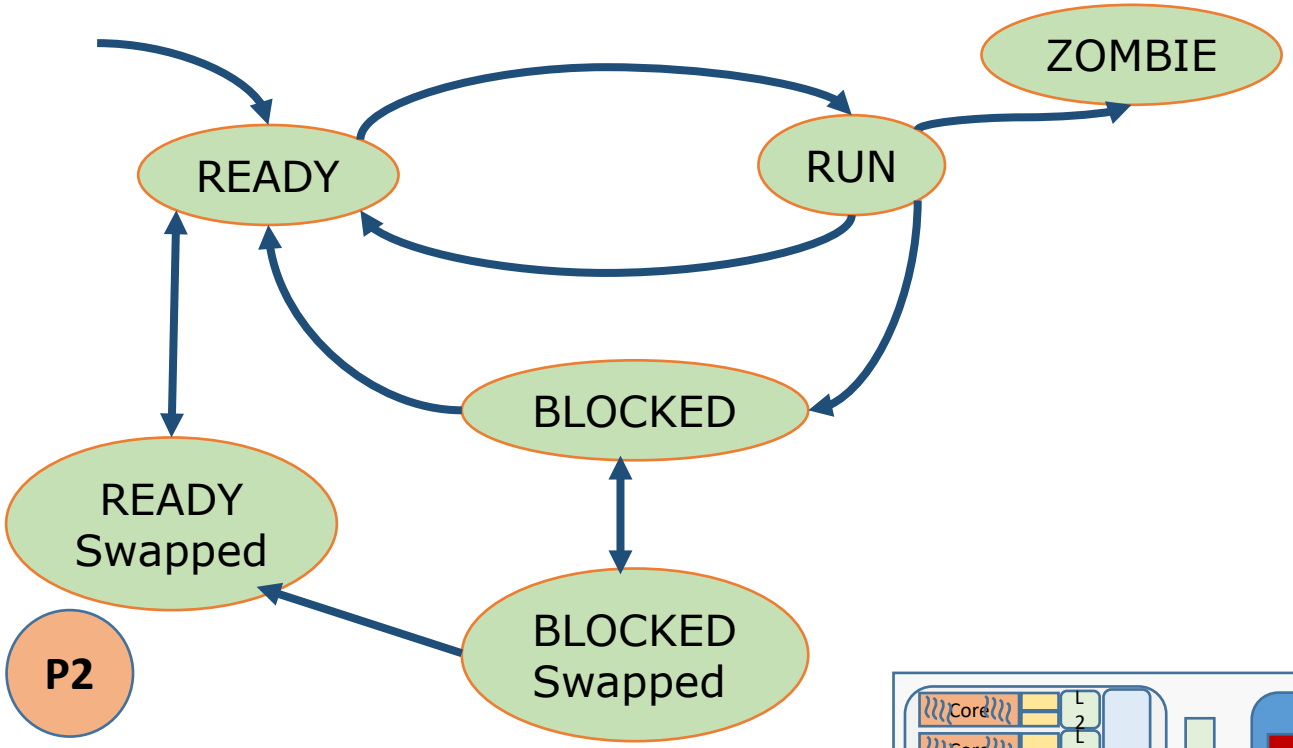
Memòria física

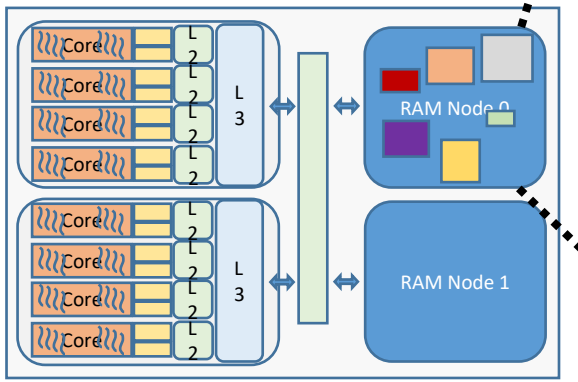


Disc

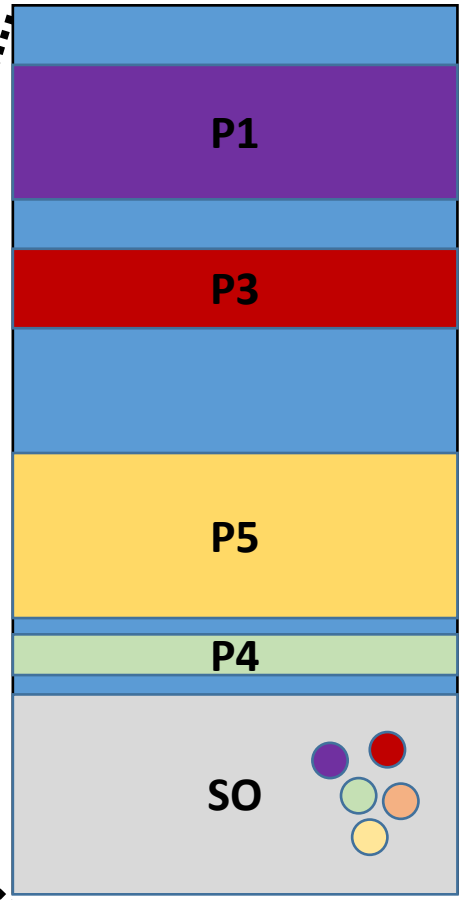


Exemple de swapping

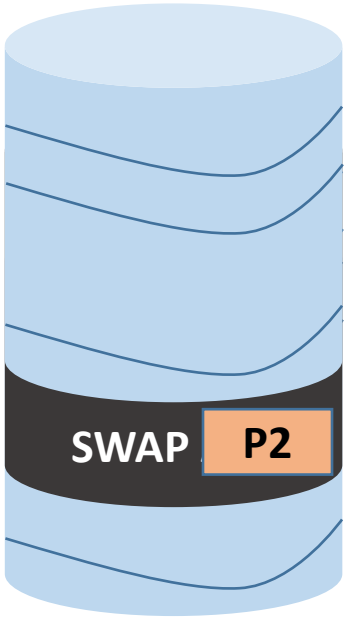




Memòria física



Disc



Impacte en el rendiment de la gestió de la Memòria

- Impacte de l'actualització de la MMU
  - En canvis de context i assignació de memòria (p.ex. llançament d'execució d'un nou programa)
  - El SO ha d'invalidar les entrades de la TLB del procés que deixa d'executar instruccions (ja no està en estat RUNNING)
- Fallos en la TLB: s'ha d'anar a memòria per accedir a la Taula de Pàgines (accés a RAM)
- Fallos de pàgina: la pàgina que s'està accedint no està carregada en memòria
  - Possible accés a disc a més d'actualització d'estructures de gestió (taula de pàgines)
- Saturació de l'ús de la memòria: queda poc (o res) espai lliure
  - Augmenta el rati de paging → més fallos de pàgina → augment rati d'accessos a disc (swap)
- Excessiu grau de multiprogramació: memòria saturada
  - Augmenta el rati de swapping → més fallos de pàgina → augment rati d'accessos a disc (swap)

Possibles Optimitzacions

- Llibreries compartides (enllaçades dinàmicament)
  - PROS: redueix el consum d'espai en disc i memòria
  - CONS: si la llibreria no s'ha carregat prèviament pot afectar negativament en el rendiment (càrrega sota demanda)
- Càrrega sota demanda: Una rutina no es carrega fins que no s'invoca
  - PROS: càrrega inicial més ràpida i menor utilització de memòria
  - CONS: durant l'execució pot baixar punctualment el rendiment
- Memòria Virtual: extensió de la càrrega sota demanda (treure lo que no és necessari)
  - PROS: augmenta el grau de multiprogramació
  - CONS: augmenta la complexitat de la gestió del Sistema de Memòria

- Hi han més optimitzacions, però que són més particulars
  - Com ara predicció en els accessos a memòria, endarrerir assignació de nova memòria física

Bibliografia

- Computer Systems – A Programmer's perspective (3rd Edition)
  - Randal E. Bryant, David R. O'Hallaron, Person Education Limited, 2016
  - https://discovery.upc.edu/permalink/34CSUC_UPC/11q3oqt/alma991004062589706711
    - Chapter 9

- Computer Organization and Design (5th Edition)
  - D. Patterson, J. Hennessy, and P. Alexander
  - http://cataleg.upc.edu/record=b1431482~S1*cat
    - Several Chapters