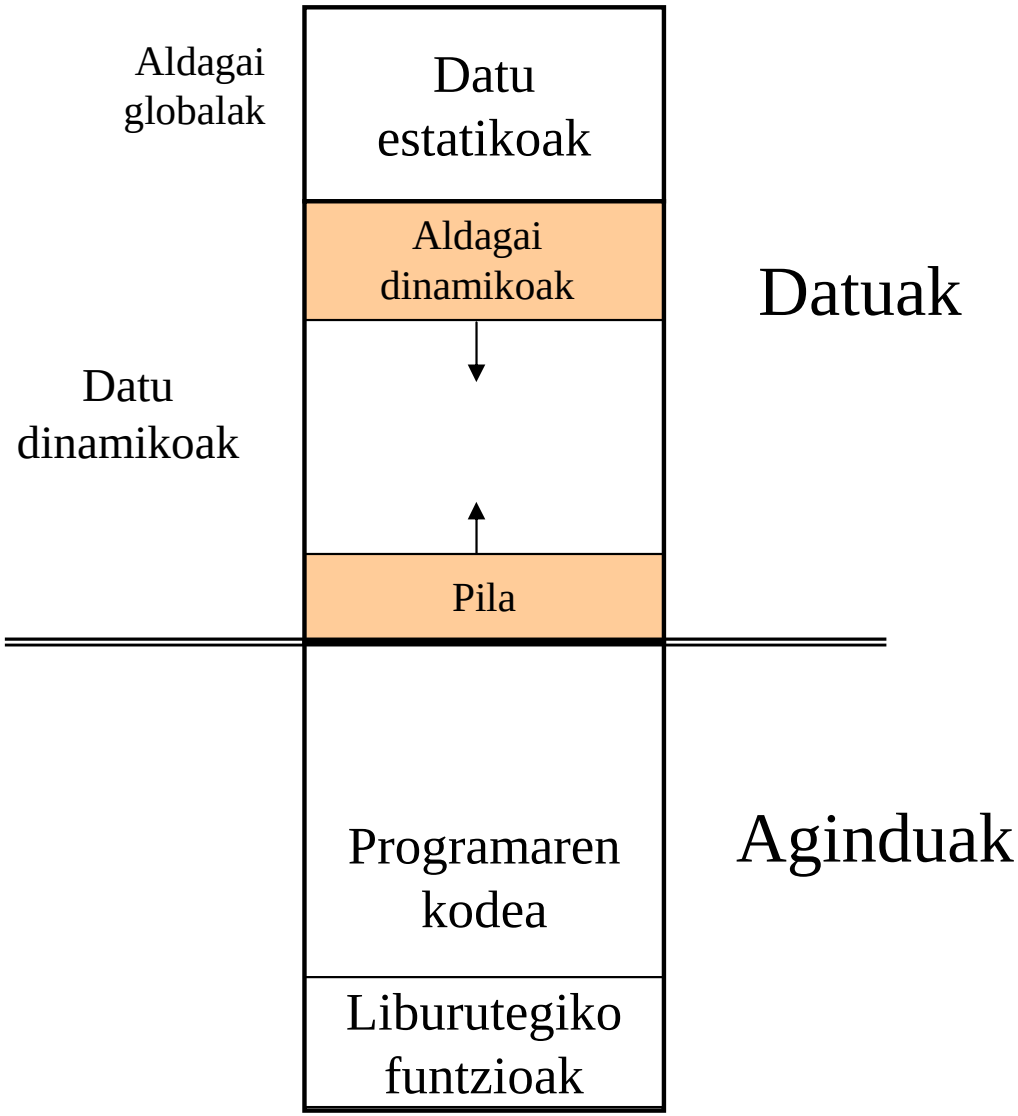
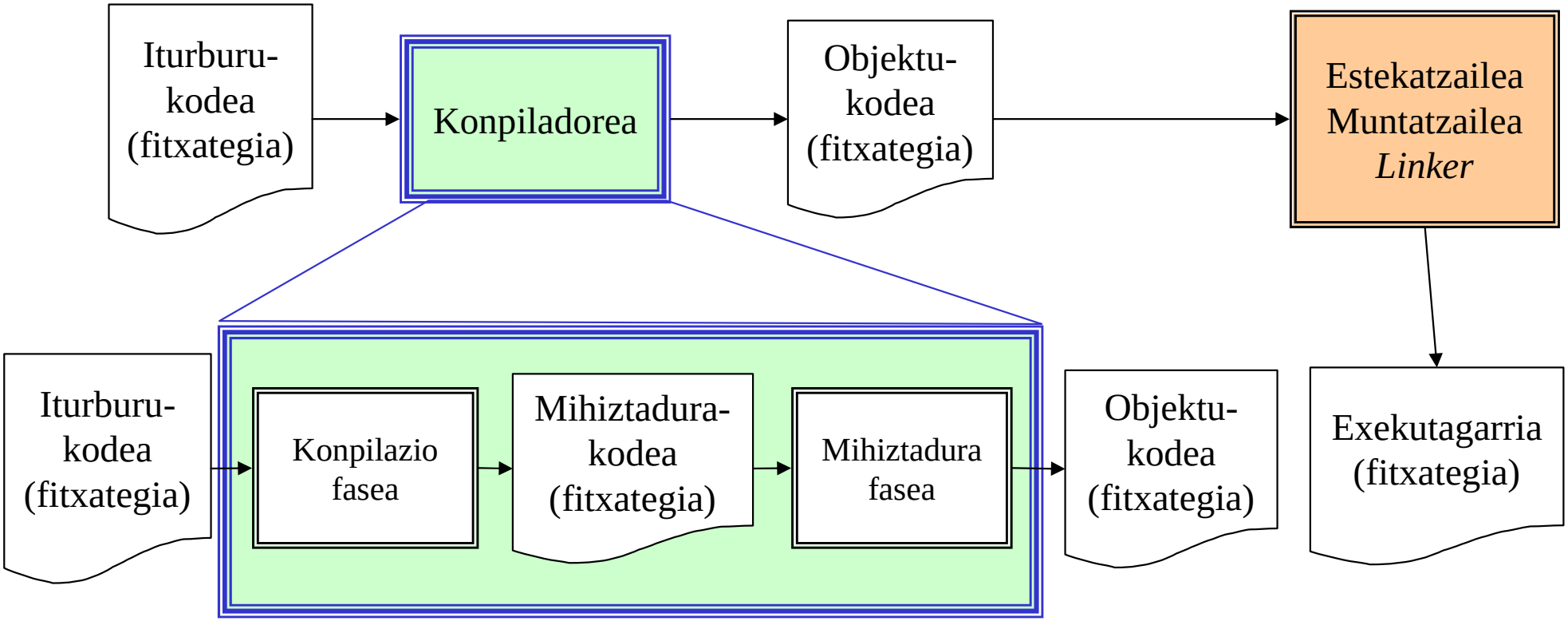
# 5. Gaia. Memoriaren kudeaketa

1. Sarrera. Programa baten egitura memorian
2. Liburutegiak eta muntaia
3. Estekatze dinamikoko liburutegiak
4. Programak memorian kokatzeko moduak
5. Programen kargarako sistema-deiak

# 5.1 Sarrera. Programa baten egitura memorian

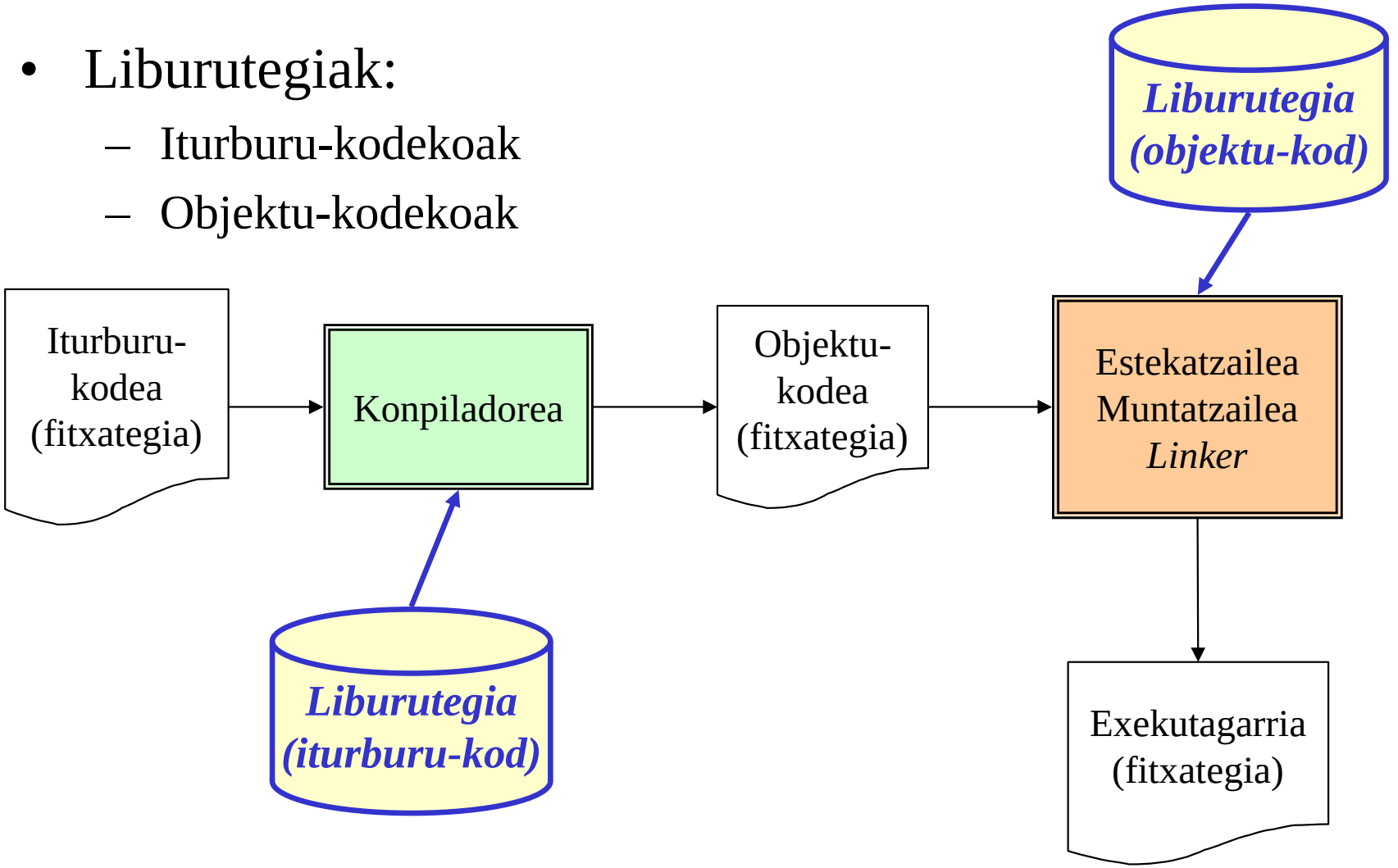


# Konpilazioa



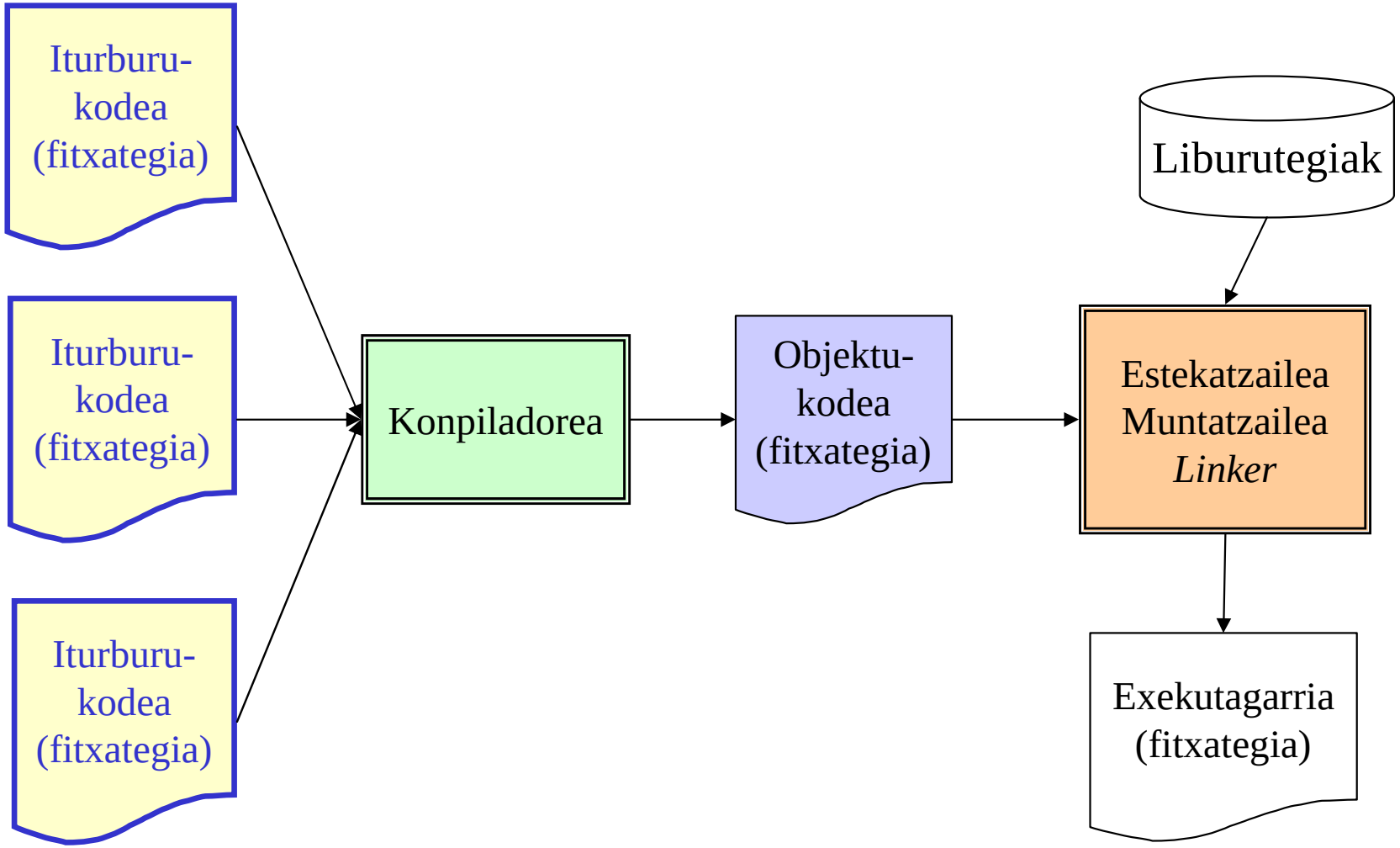
# Konpilazioa

- Liburutegiak:
  - Iturburu-kodekoak
  - Objektu-kodekoak



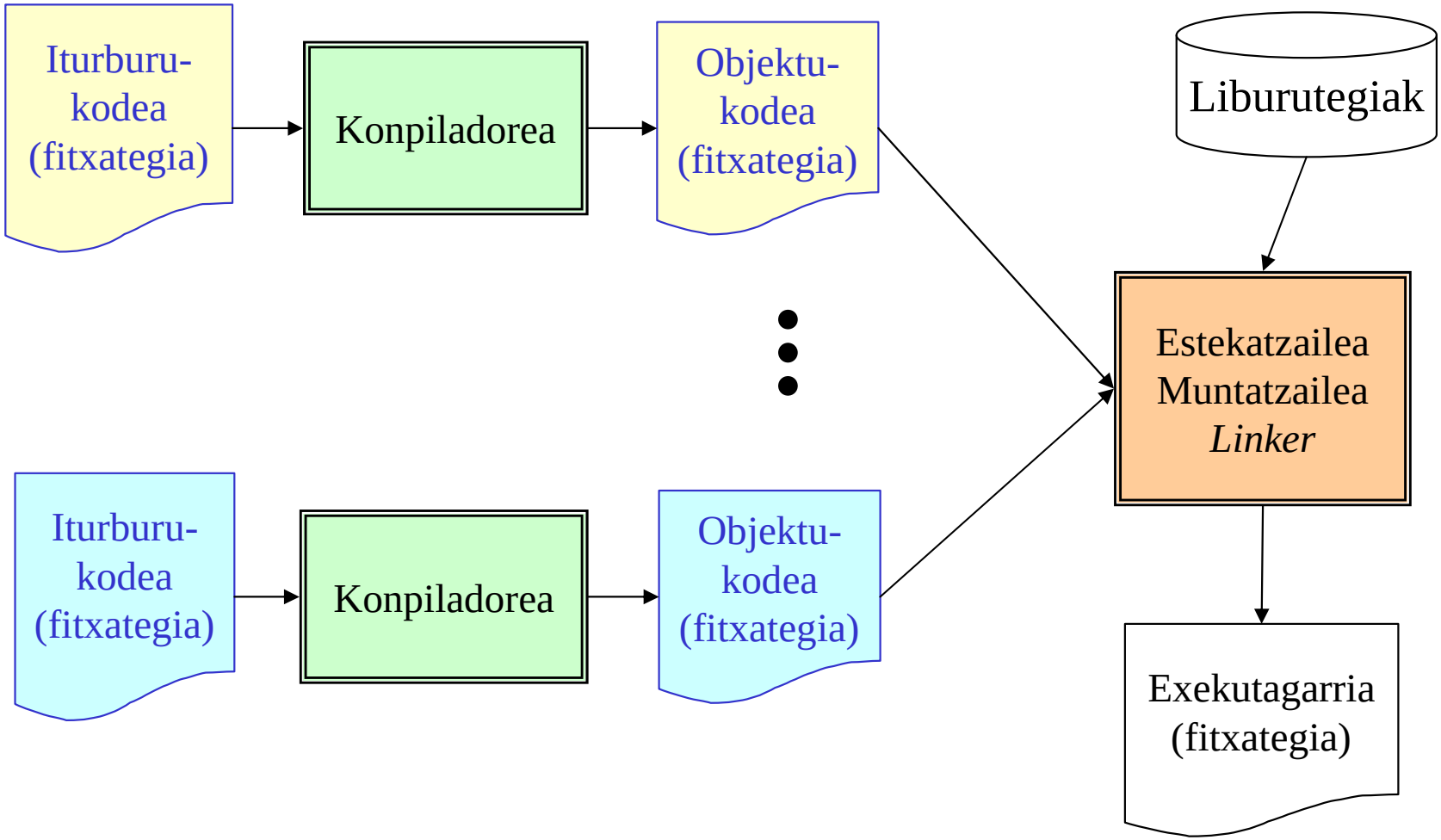
# Konpilazioa

- Hainbat iturburu ⇨ Objektu modulu bakarra



# Konpilazioa

- Hainbat iturburu ⇨ Hainbat objektu modulu



# 1. Ariketa. Konpilazio faseak gcc konpiladorearekin

1. Linuxeko *man* laguntza kontsultatuta, aztertu nola ikus daitekeen konpilazioaren fase desberdinak
2. Probatu **`kopiatu.c`** programaren konpilazioa
3. Konpilazioaren fase bakoitza komentatu
4. Saia zaitez beste konpiladore batekin probatzen

# Goi-mailako lengoaiak (GML)

- Konpiladorea
  - Goi-mailako lengoaiako ***agindu 1*** => makina-lengoaiako ***n agindu***
- Itzulpena + Optimizazioa
  - ✚ eraginkorragoa
  - ✚ trinkoa
- Arazoa eragigaien helbideak itzultzean: aginduak eta aldagaiak zein memoria-helbidetan kokatu?
  - Konpiladoreak hasierako karga-helbidea ezagutzen du (***Konpilazio-denborako birkokapena***).
  - Konpiladoreak hasierako karga-helbidea 0 helbidea dela suposatzen du. Ondoren, programa memorian kargatzerakoan (kargatzaileak) karga-helbide errealarekin birkokatzen ditu helbide guztiak (***Karga-denborako birkokapena***).
- ***Birkokapen estatikoa***
  - Birkokapen-taula

# Iturburu-kodea. Konpilazioa

Iturburu-programa
goi-mailako lengoaian (C)

```
...
int Globala1, x, Emaitza;

main()
{
   Globala1 = 35;
   x = 3;
   Emaitza = Globala1 + x*x;
   idatzi(Emaitza);
}

void idatzi(int E)
{
   printf("Emaitza: %d izan da\n", E);
}
```

*Helbide logikoak*

Konpiladorea

Mihiztadura-kodea

```
0:    Globala1:
4:    x:
8:    Emaitza:
      PROC main
12:   PUSH r30          ;; main
16:   MOV r30, r31
20:   MOV r1, 35
24:   ST r1, @Globala1
28:   MOV r1, 3
32:   ST r1, @x
36:   LD r2, @Globala1
40:   LD r3, @x
44:   MULT r4, r3, r3
48:   ADD r5, r2, r4
52:   ST r5, @Emaitza
56:   PUSH r5
60:   CALL idatzi
...

128:  RET

      PROC idatzi
132:  PUSH r30   ;; idatzi
...
252:  RET
```

# Bigarren fasea

Mihiztadura-kodea

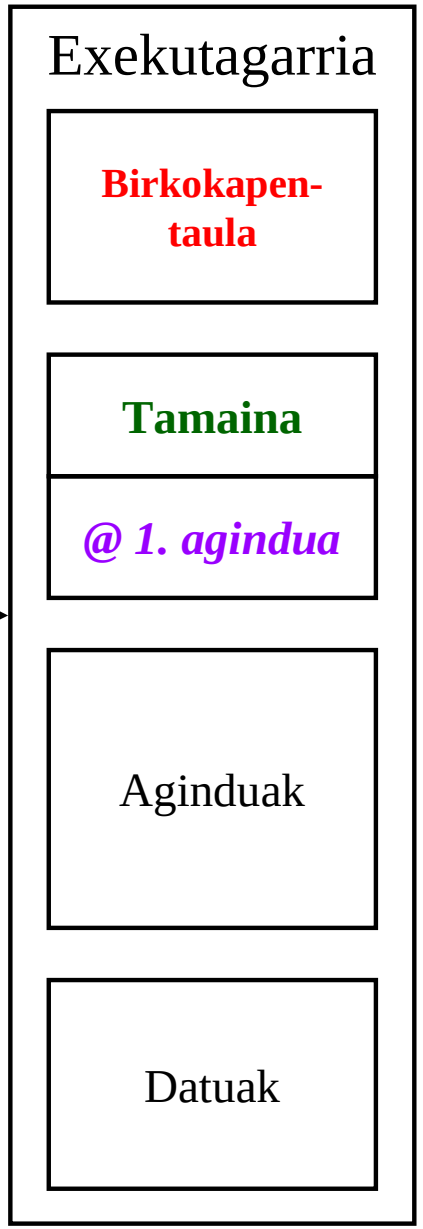
```
0:      Globala1:
4:      x:
8:      Emaitza:
        PROC main
12:     PUSH r30   ;; main
16:     MOV r30, r31
20:     MOV r1, 35
24:     ST r1, @Globala1
28:     MOV r1, 3
32:     ST r1, @x
36:     LD r2, @Globala1
40:     LD r3, @x
44:     MULT r4, r3, r3
48:     ADD r5, r2,r4
52:     ST r5, @Emaitza
56:     PUSH r5
60:     CALL idatzi
...

128:    RET

        PROC idatzi
132:    PUSH r30   ;; idatzi
...
252:    RET
```

Mihiztatzailea

Exekutagarrien formatua:
a.out,
COFF,
***ELF***,
...



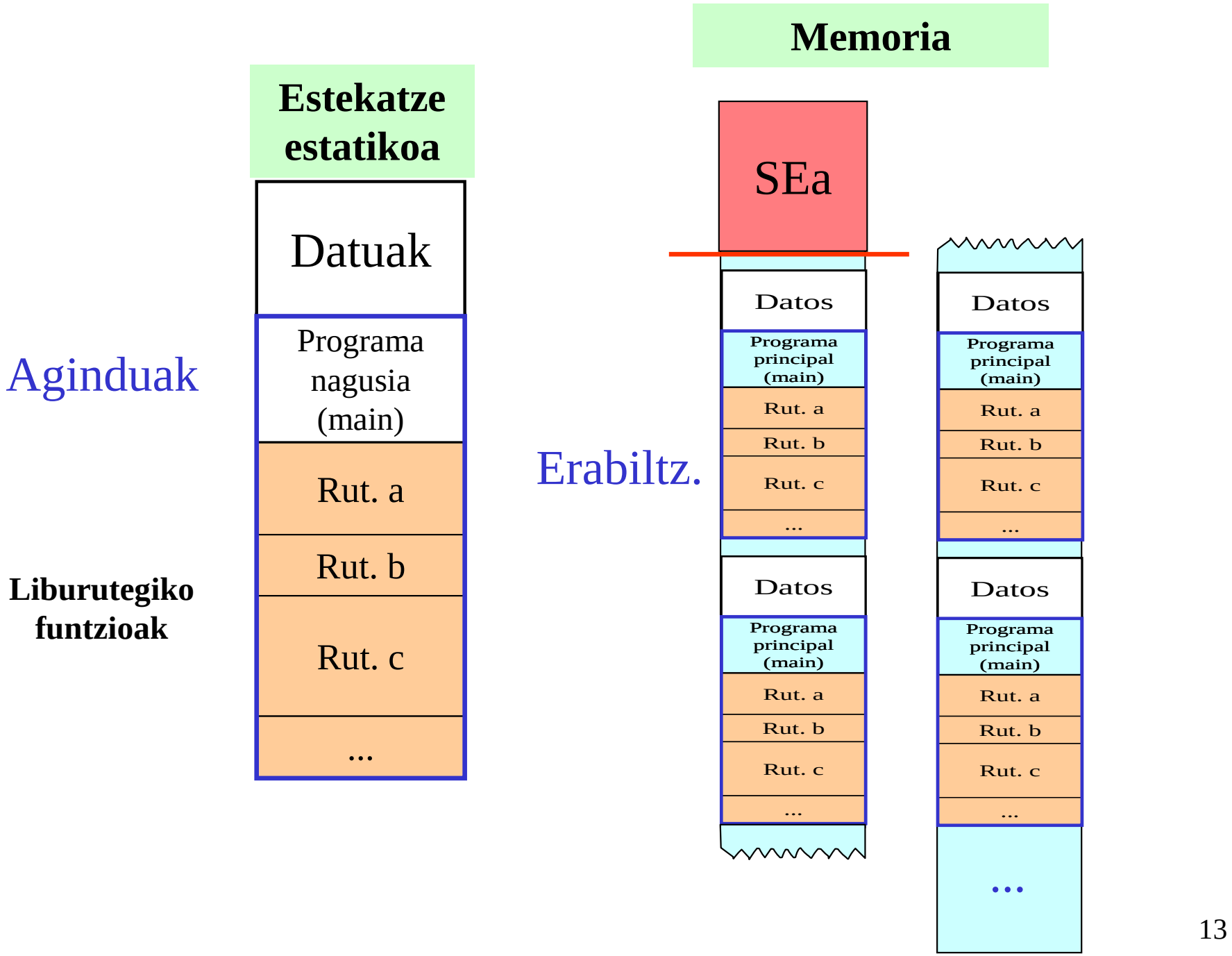
# Fitxategi exekutagarrien formatua

- Linuxen hainbat formatu daude:
  - ELF (Executable and Linking Format)
    - Unix System Laboratories garatua
    - http://www.skyfree.org/linux/references/ELF_Format.pdf
    - Hainbat SEendako estandarra
  - a.out (Assembler OUTput format)
    - Linuxeko bertsio zaharrak
- Beste SEetako formatuak:
  - MS-DOSeko EXE formatua
  - Unix-BSDko COFF formatua

# 5.2 Liburutegiak eta muntaia. Errutinen berrerabilpena

- Helburuak: birprogramazio lana minimizatu, erroreak minimizatu
- Goi-mailako lengoaiako funtzioen liburutegiak (iturburu-liburutegiak)
  - Ez dira birprogramatu behar. Baina birkonpilatu behar dira
- Objektu-kodeko funtzioen liburutegiak (dagoeneko konpilatuak)
  - Konpilazio banatua
  - Iturburu-programan agertzen ez diren errutinen aipamenak agertzen dira
  - Errutinek programa deitzaileak dauzkan aldagai globalak erreferentzia ditzakete
  - EBATZI GABEKO ERREFERENTZIEN arazoa
- Muntatzailea – Estekatzailea (*Linker*)
  - Ebatzi gabeko erreferentziak konpontzen ditu, exekutagarria eraikiz
  - Goi-mailako lengoai desberdinen konbinaketa
    - Egitura bereko objektu-kodea
    - Teknika berdinak errutinei parametroak pasatzeko eta aldagai globalak atzitzeko
- ***Estekatze-denborako birkokapena***

# Estekatze estatikoa



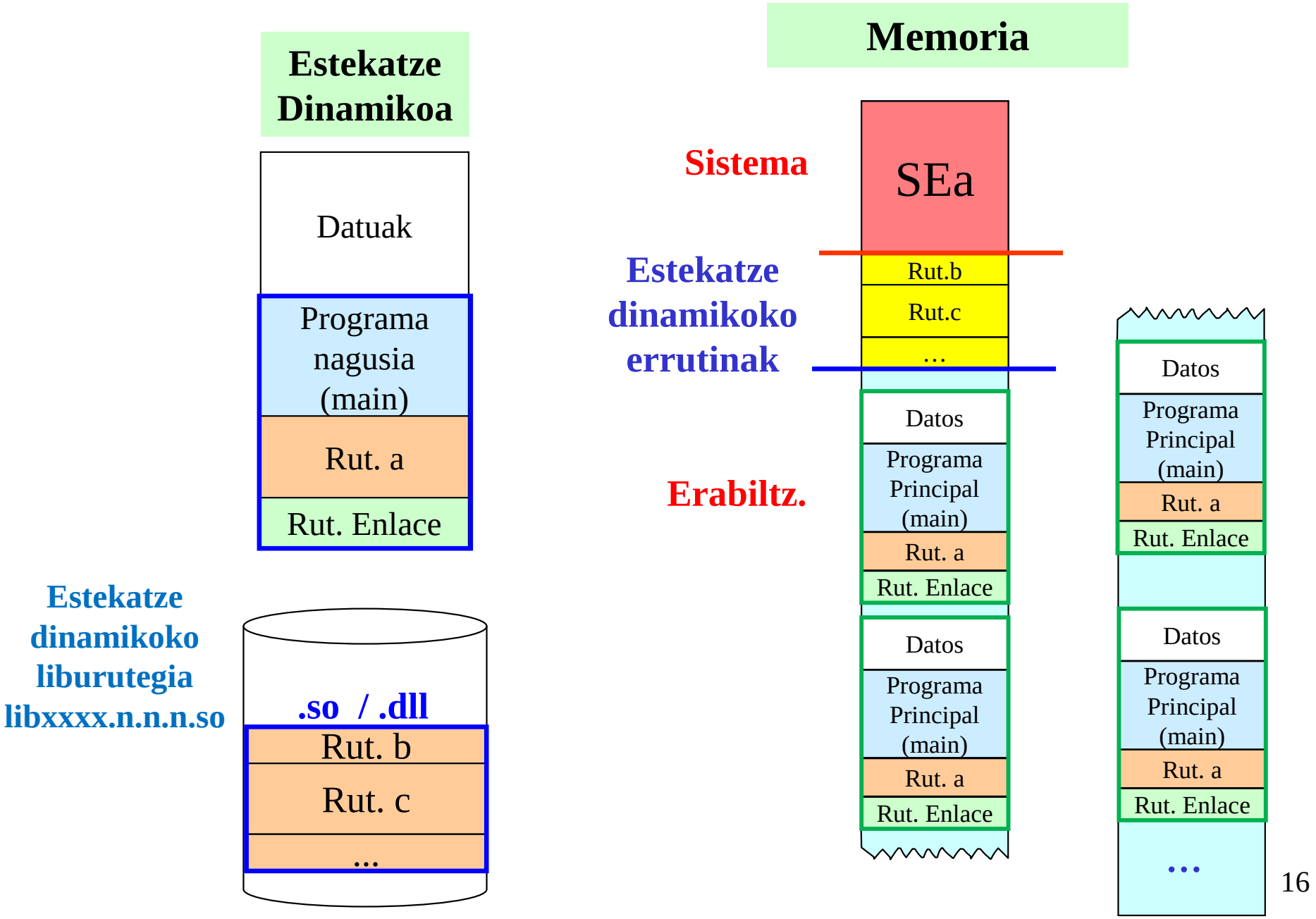
# Estekatze estatikoa

- Liburutegi estatikoak:
  - Linkedizioa: programaren objektu moduluak eta liburutegiak estekatzea
  - Fitxategi exekutagarria: modulu guztiak barneratzen ditu
- Estekatze estatikoaren desabantailak:
  - Liburutegiko funtzioen kodea errepikatuta exekutagarri desberdinetan
  - Fitxategi exekutagarri handiak (diskoan eta memorian)
  - Liburutegiko funtzioen kopia anitz memorian
  - Liburutegien eguneraketa: exekutagarri guztiak berriro linkatu behar
- Hobekuntza: **estekatze dinamikoa**
  - Funtzio erabilienen kodea ez errepikatu (memorian behin)
  - Estekatze dinamikoko liburutegiko funtzioen kodea ez dago fitxategi exekutagarrietan
  - Fitxategi exekutagarri txikiagoak (diskoan eta memorian)

# 5.3 Estekatze dinamikoko liburutegiak

- Sistema Eragilearen errutinetarako pentsaturiko errutinen berrerabilgarritasuna edozein motako errutinetara zabaldu
- Espazio-aurrezpena, programek gutxiago okupatzen dute, bai memorian eta bai diskoan
- Estekatze dinamikoko liburutegia. Programetan mota honetako funtzioei deitzeko, errutina berezia erabiltzen da
  - Estekatzaileak programa exekutagarrian errutinak zuzenean sartu beharrean sistema-dei berezi bat ipintzen du (estekatze-errutina)
  - Exekuzio-denboran kargatzen dira, soilik kargaturik gabe badaude
    - programa kargatzen denean / funtzioari deitzen zaionean
- Programak aldatu/eguneratu daitezke birkonpilatu gabe
- SEaren errutinetara aplikatu daiteke (moduluak)
  - SEaren zati desberdinak aldatu daitezke sistema gelditu gabe
- Memoria fisikoa baino handiagoak diren programak exekutatu daitezke (alegiazko memoria mekanismoa)

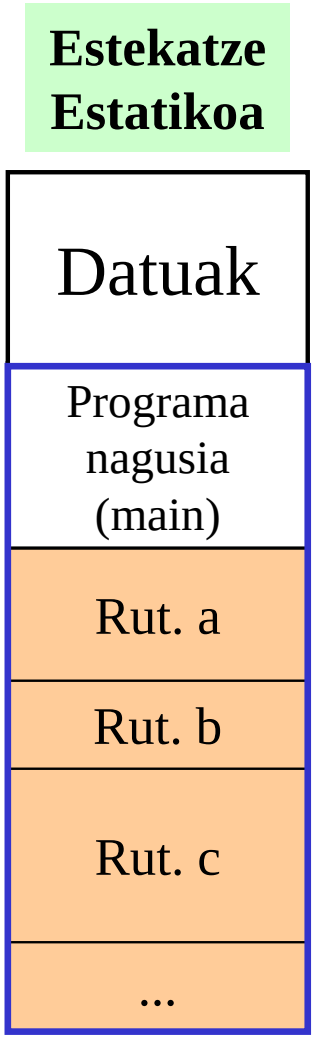
# Estekatze dinamikoa

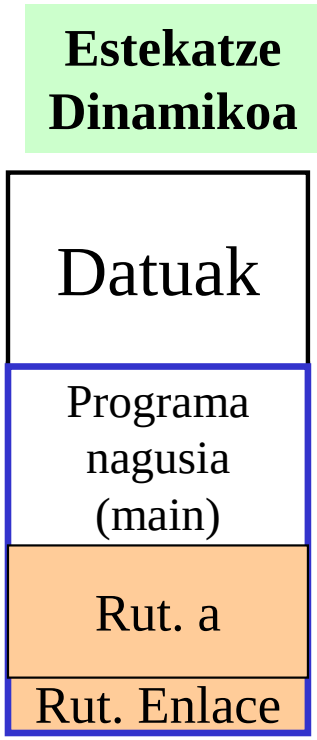


# Estekatze estatikoa eta dinamikoa

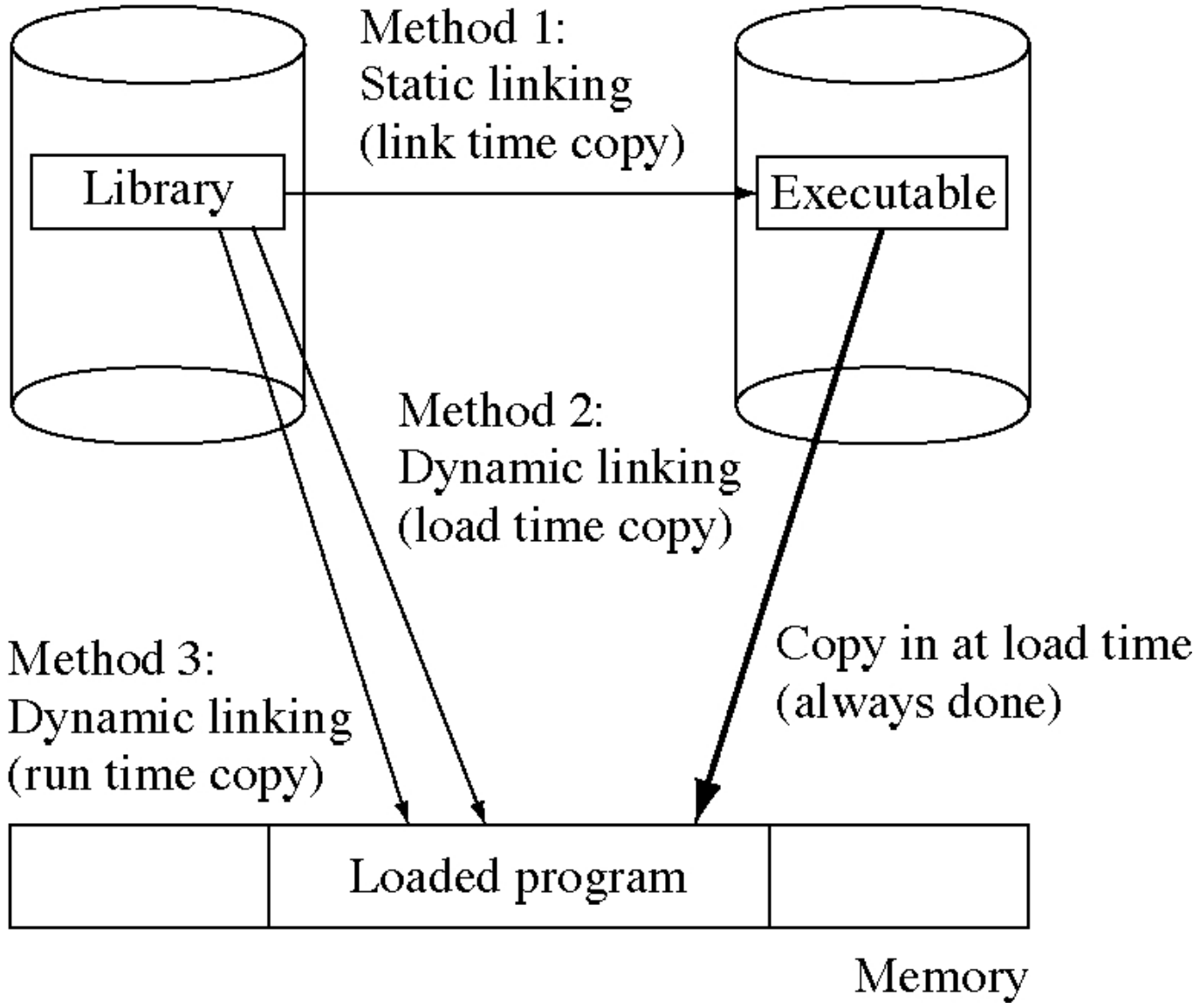
Aginduak

**Liburutegiko funtzioak**



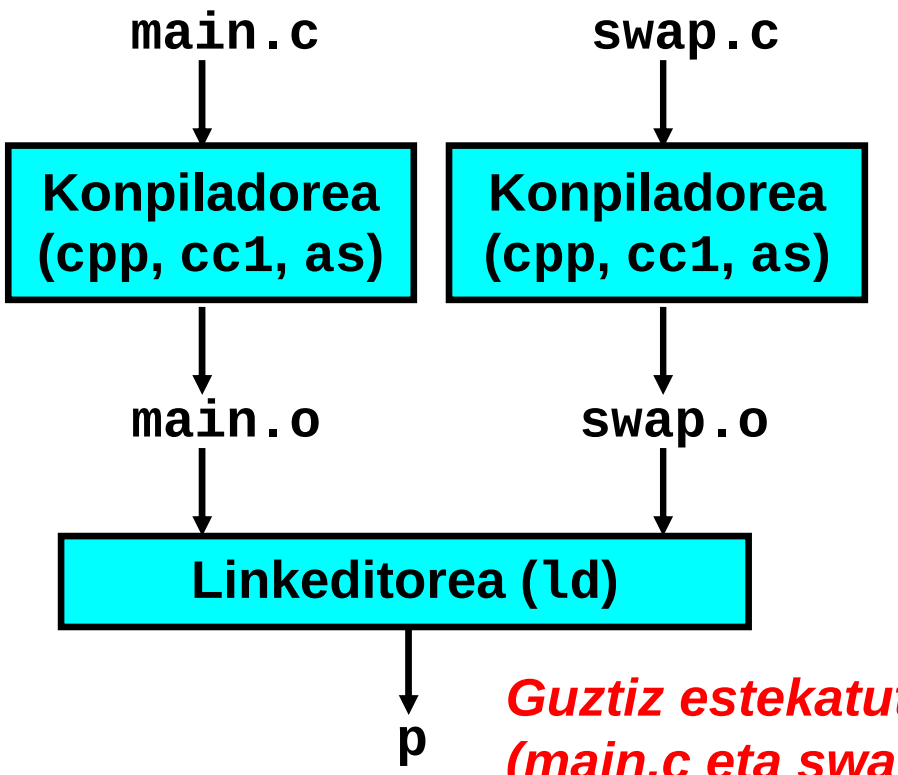
# Estekatze estatikoa eta dinamikoa



# Estekatze dinamikoa

- Abantailak:
  - Fitxategi exekutagarriak txikiagoak (diskoan espazioa aurrezten da)
  - Liburutegi berdina erabiltzen duten hainbat programa memorian kargatuta izan ezkero, memoria gutxiago okupatzen dute
  - Aplikazioak eguneratu daitezke osorik birkonpilatu gabe (soilik liburutegia)
  - Memoria fiskoa baino handiagoak diren programak exekutatu daitezke (liburutegien kudeaketa egokia bada)
  - SEra aplikatuz, SEa bera aldatu daiteke geldiarazi gabe
- Eragozpenak:
  - Dei mekanismoa (zertxobait) motelagoa
  - Liburutegien kudeaketa konplexuagoa. Bertsioen kudeaketa beharra
  - Aplikazioak instalatzerakoan, estekatze dinamikoko liburutegien menpekotasunak kontuan hartu behar dira

# Estekatze estatikoa

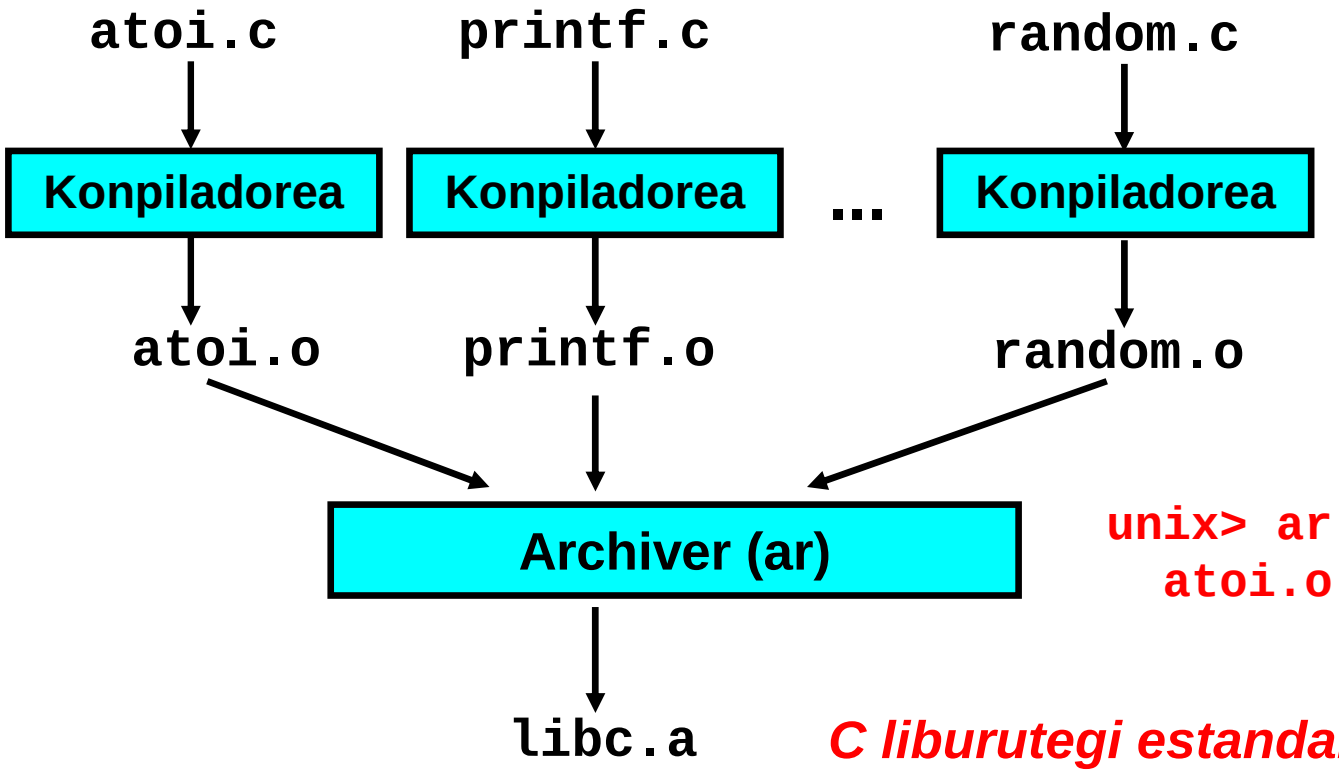



*Iturburu fitxategiak*

*konpilatutako Objektu fitxategiak*

*Guztiz estekatutako exekutagarri kodea (main.c eta swap.c fitxategietan definitutako funtzio guztientzako kodea eta datuak barneratzen ditu)*

# Liburutegi estatikoen sorrera



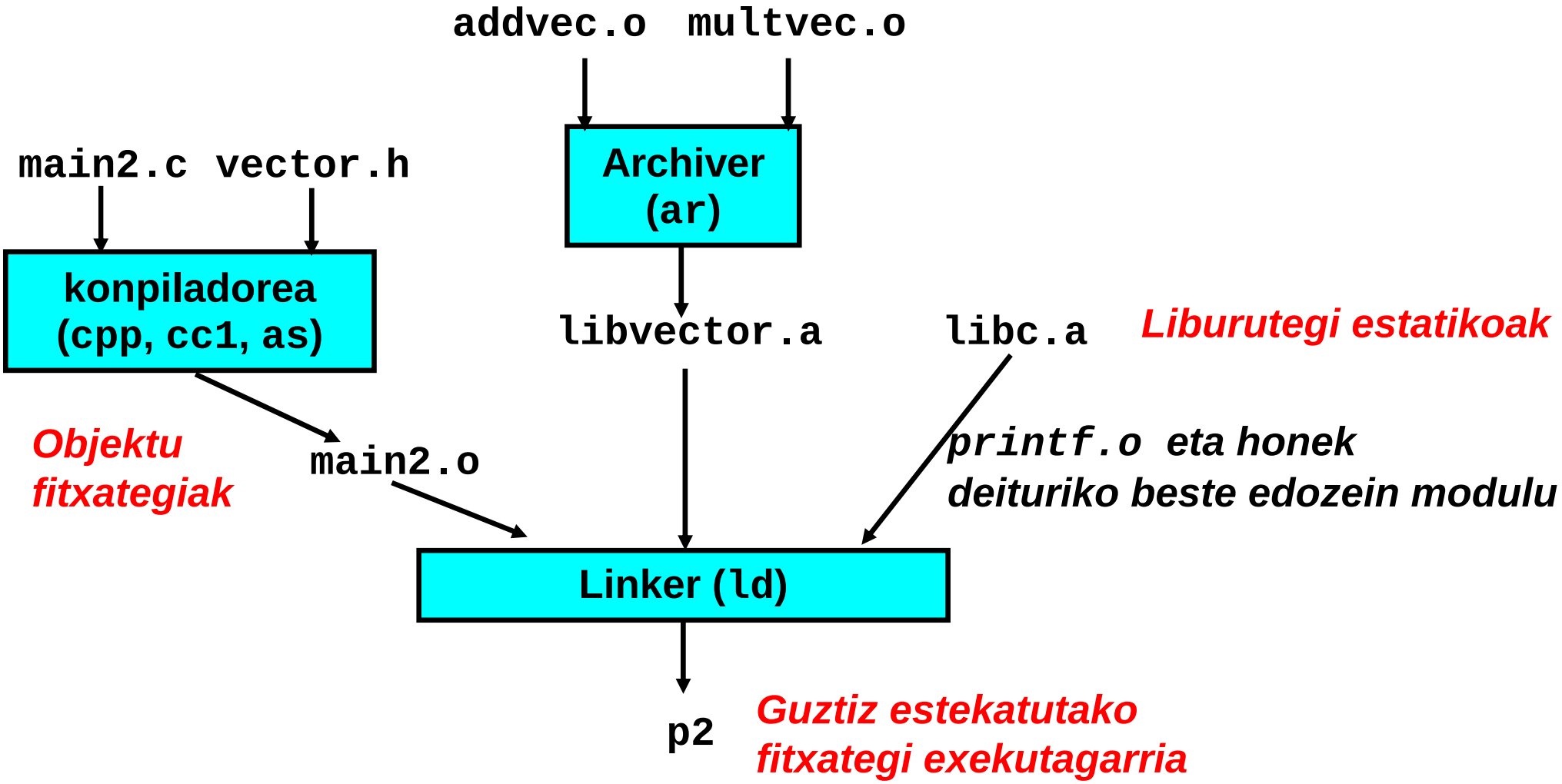

```
unix> ar rs libc.a \
  atoi.o printf.o … random.o
```

*C liburutegi estandarra*

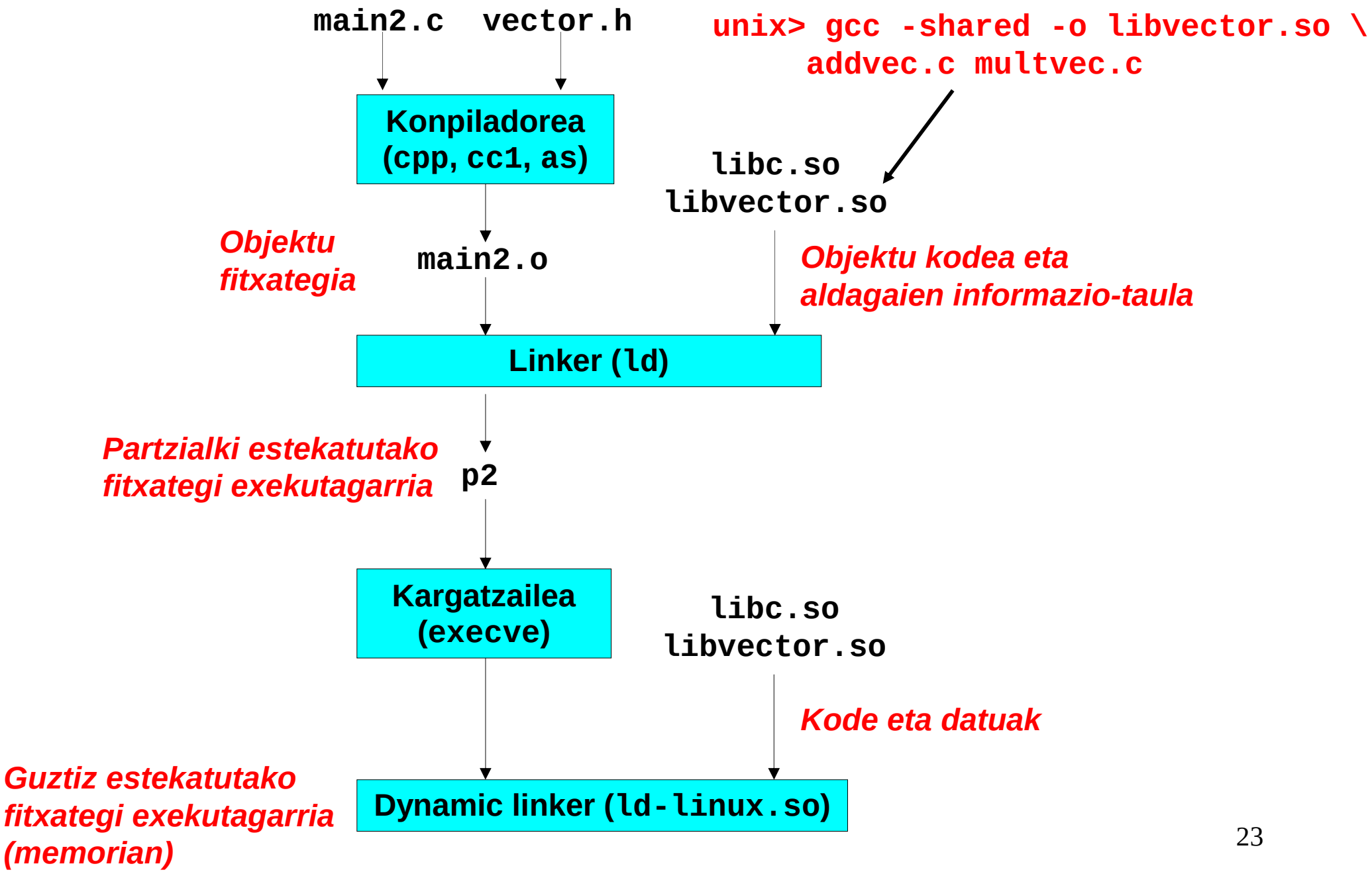
***Archiver* programak eguneraketa inkrementalak onartzen ditu:**

- **Birkonpilatuz liburutegian aldatu beharreko .o fitxategia**

# Estekatzea liburutegi estatikoekin



# Estekatze dinamikoa (karga garaian)



# 2. Ariketa. Liburutegi estatikoak eta dinamikoak

1. Bilatu Interneten “*Creating a shared and static library with the gnu compiler…*” artikulua

2. Sortu liburutegi estatiko eta dinamiko bana **`pi_kalkulatu.c`** fitxategiarekin

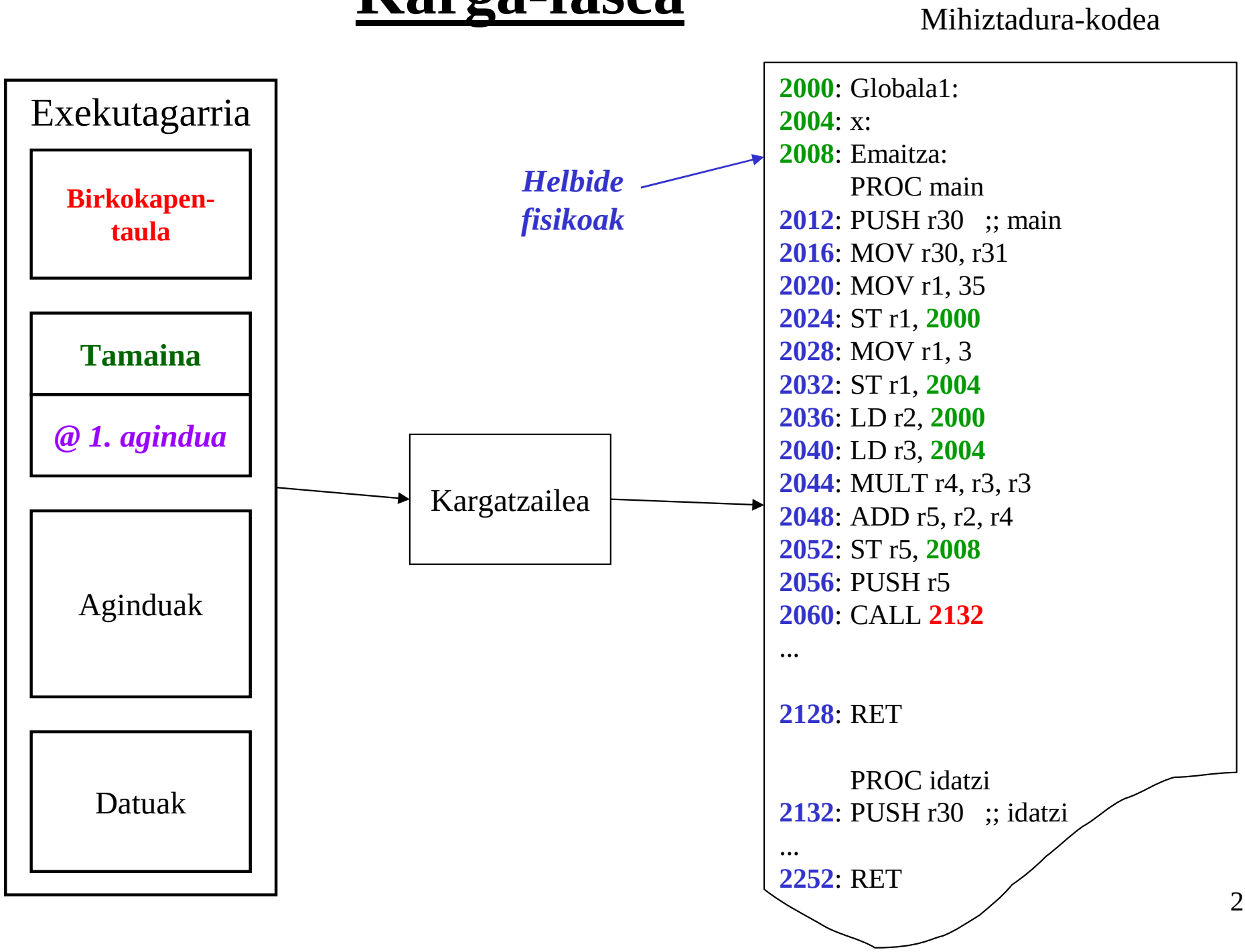
3. Probatu aurreko bi liburutegiak **`pi_test.c`** programaren bidez (programaren bi bertsio sortuz). Konparatu fitxategi exekutagarrien tamainak

4. Aztertu aurreko bi programen exekuzio denborak UNIXeko *time* tresnaren bidez, 10000000 eta 300000000 balioak erabiliz. Zer ondorio atera dezakezu?

# 5.4 Programak memorian kokatzeko moduak. Karga eta exekuzio faseak

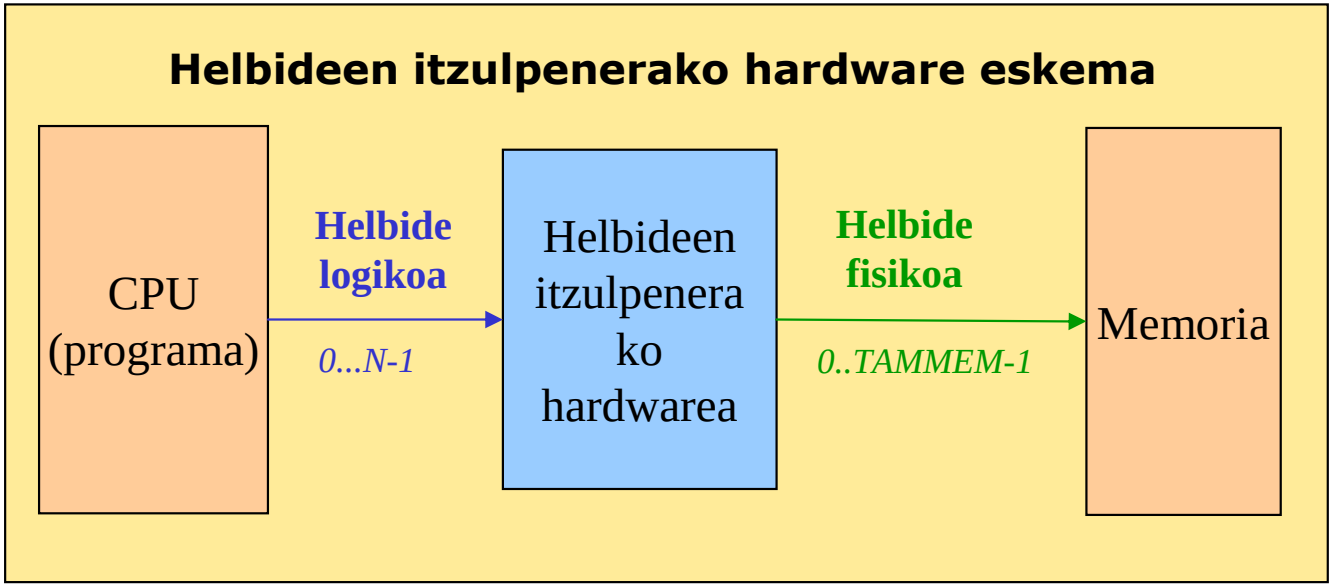
- Kargatzailearen betebeharrak:
  1. Fitxategi exekutagarria irakurri
  2. Memorian kokatu:
     - Memorian behar beste tamainako zati libre eta jarraitua bilatu
     - Programaren kodea kopiatu
     - Hasieratutako datuak kopiatu
     - Datu dinamikoentzako tokia erreserbatu (pila, memoria dinamikoa)
  3. Programako lehen agindua abiatu:
     - CPU-aren erregistroak hasieratu (PC, PSW, SP...)
- SEak memoriaren erabileraren kudeaketa egin behar du:
  - Libre dagoen memoria jakin
  - Programa bakoitzak okupatzen duen memoria tartea jakin
  - Programa bat bukatzean, bere memoria askatu behar da

# Karga-fasea


Exekutagarria
Birkokapen-taula
Tamaina
@ 1. agindua
Aginduak
Datuak
Kargatzailea
Helbide fisikoak
Mihiztadura-kodea
2000: Globala1:
2004: x:
2008: Emaitza:
PROC main
2012: PUSH r30 ;; main
2016: MOV r30, r31
2020: MOV r1, 35
2024: ST r1, 2000
2028: MOV r1, 3
2032: ST r1, 2004
2036: LD r2, 2000
2040: LD r3, 2004
2044: MULT r4, r3, r3
2048: ADD r5, r2, r4
2052: ST r5, 2008
2056: PUSH r5
2060: CALL 2132
...
2128: RET
PROC idatzi
2132: PUSH r30 ;; idatzi
...
2252: RET

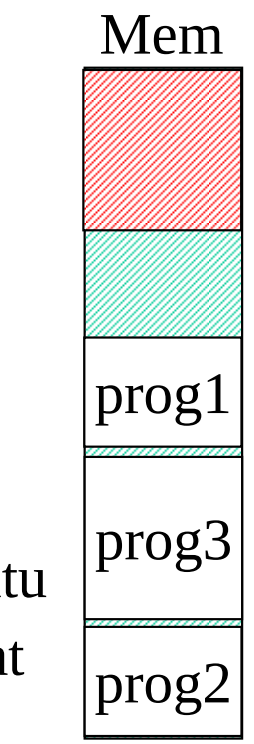
# Programak memorian kokatzeko moduak

- Programa bat baino gehiago memorian
  - EZ JARRAIAN
  - EZ OSORIK
- Birkokapen estatikoa *versus* birkokapen dinamikoa

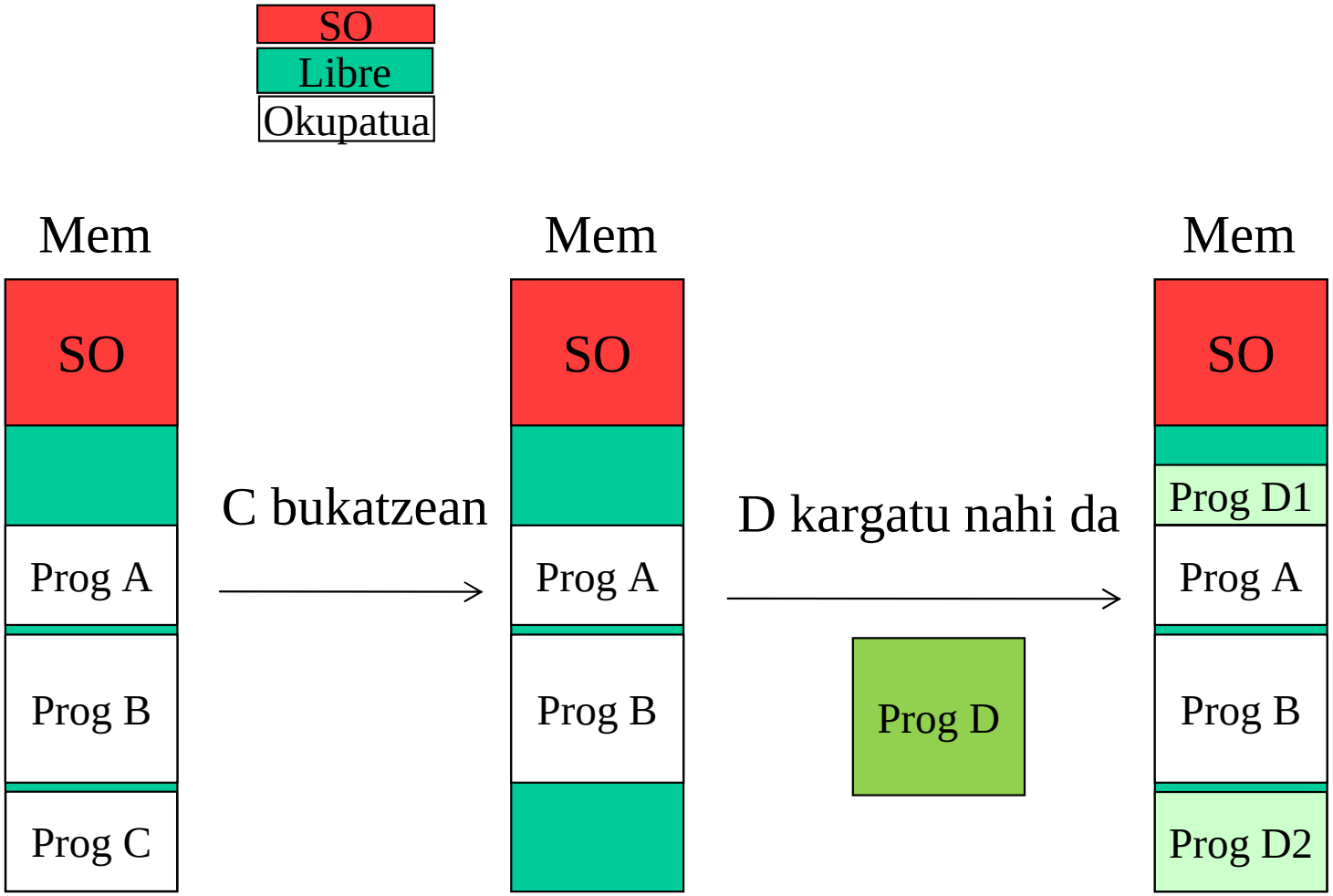


# Programa bat baino gehiago memorian

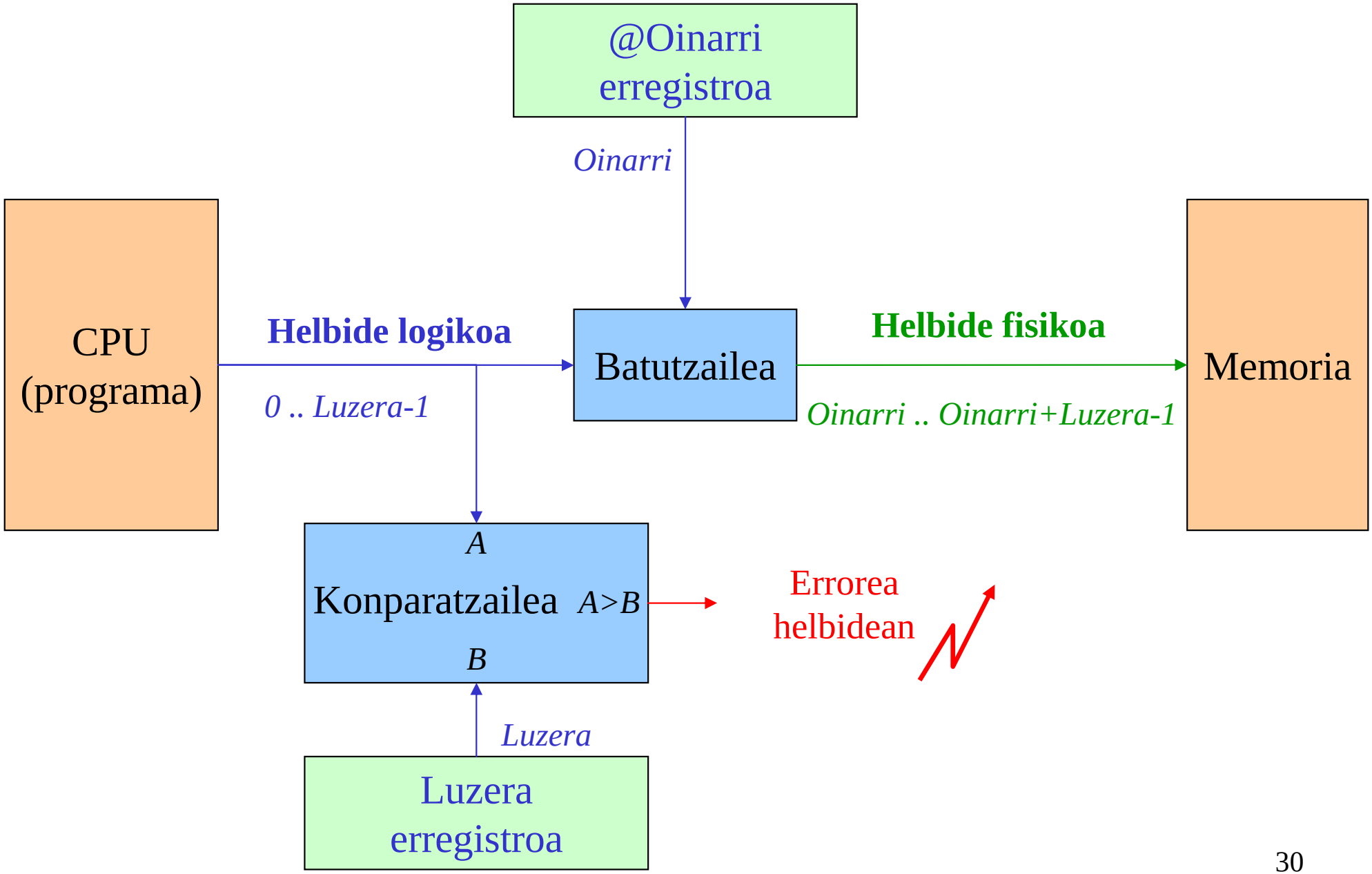
- Orain arte programa bakarra memorian, OSORIK eta JARRAIAN
- Ordenagailuek gero eta memoria gehiago
  - Gehien erabiltzen diren programak (KIa, editorea, konpiladorea, estekatzailea, kargatzailea, etab.) memorian egoiliar
- Ezinezkoa da programa guztiak memorian egoiliar mantentzea
  - Maiz erabiltzen diren programek memoria espazio bat konpartitzea
  - Memoria espazioaren kudeaketa
  - Ordezkatze-politikak
- *exekutatu_programa* errutina: programa memorian dagoen egiaztatu
- Oraingoz exekuzioa sekuentziala dela suposatuko da (programa bat amaitu arte ezin da beste bat hasi)
- Programa guztiak erabiltzailearen helbide-esparruan daude (babesik gabeko esparrua). Memorian programa bat baino gehiago egotean, beste programen nahigabeko atzipenaz babesteko beharra agertzen da
- Babeserako hardware berezia
- ***Birkokapen DINAMIKOA*** (exekuzio-denboran)



# Memoriaren fragmentazioa

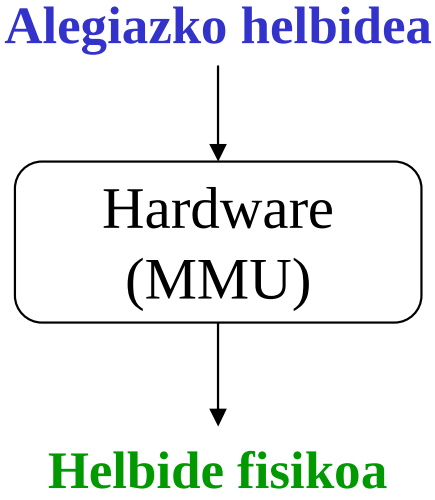


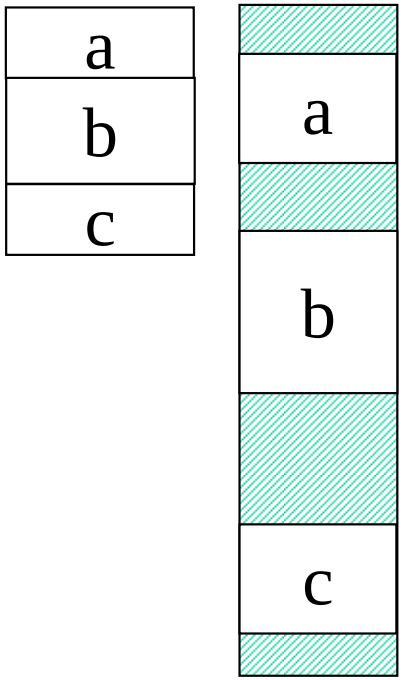
# Memoriaren babeserako hardware-eskema

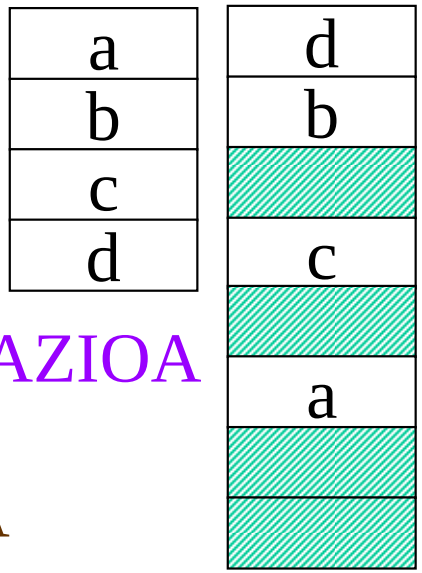


# Programa EZ JARRAIAN eta EZ OSORIK

- EZ JARRAIAN

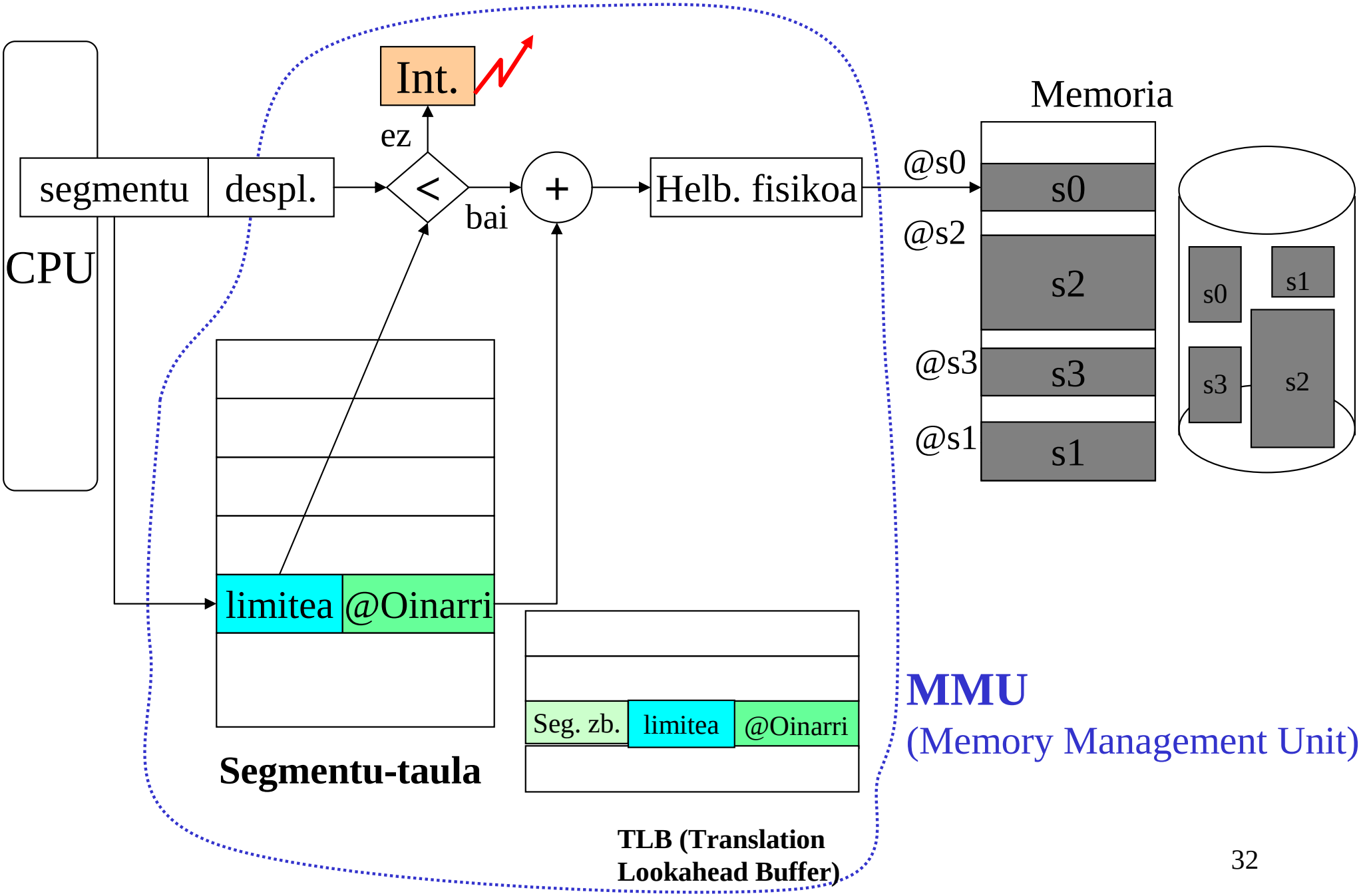


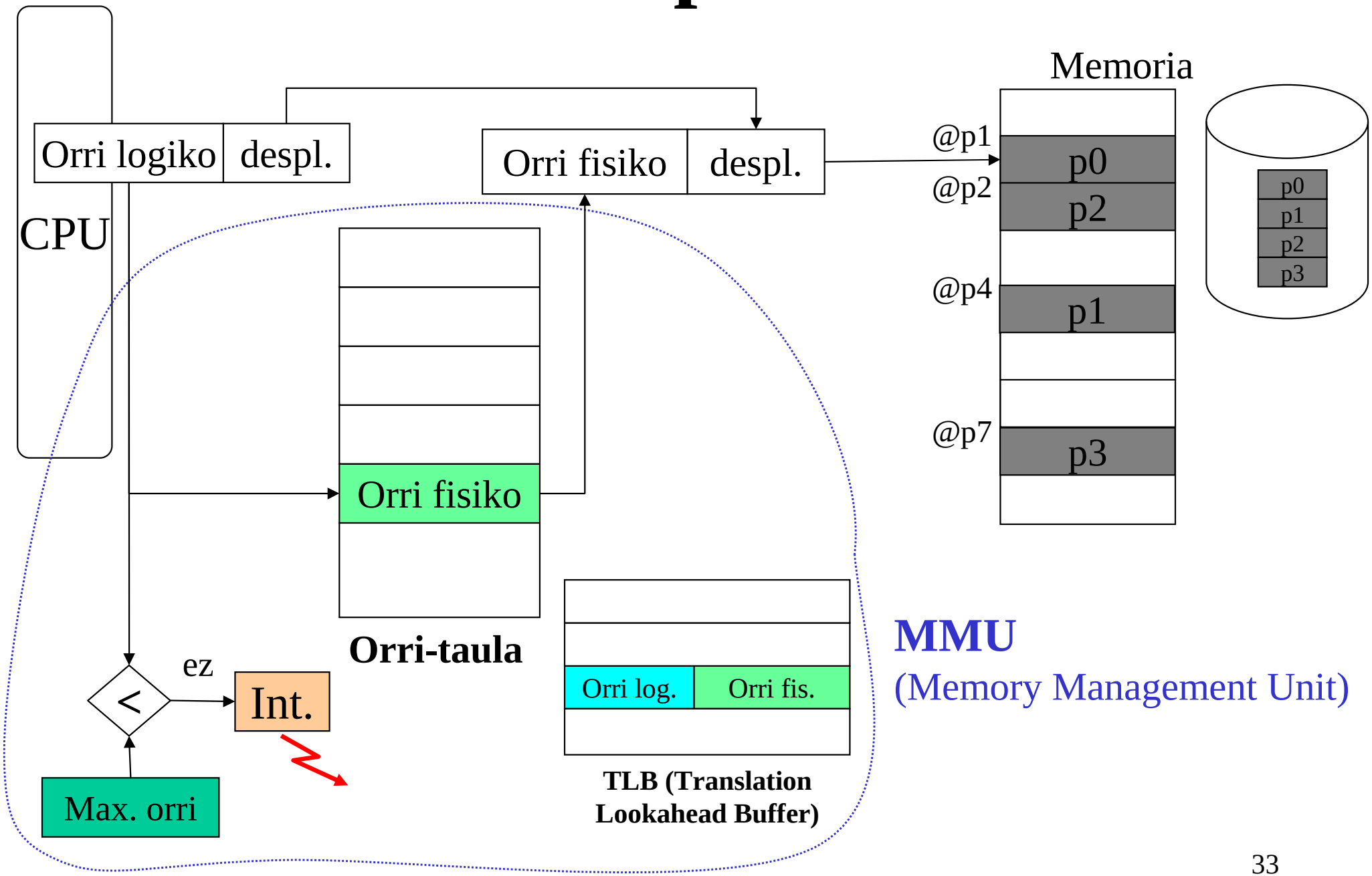

- Programa zatitan banatu. Bi teknika:
  - Tamaina desberdineko zatiak: SEGMENTAZIOA
    Konpiladoreak egiten du
  - Tamaina berdineko zatiak: ORRIKAPENA
    Sistema Eragileak egiten du

# Segmentazioa




**MMU**
(Memory Management Unit)

# Orrikapena

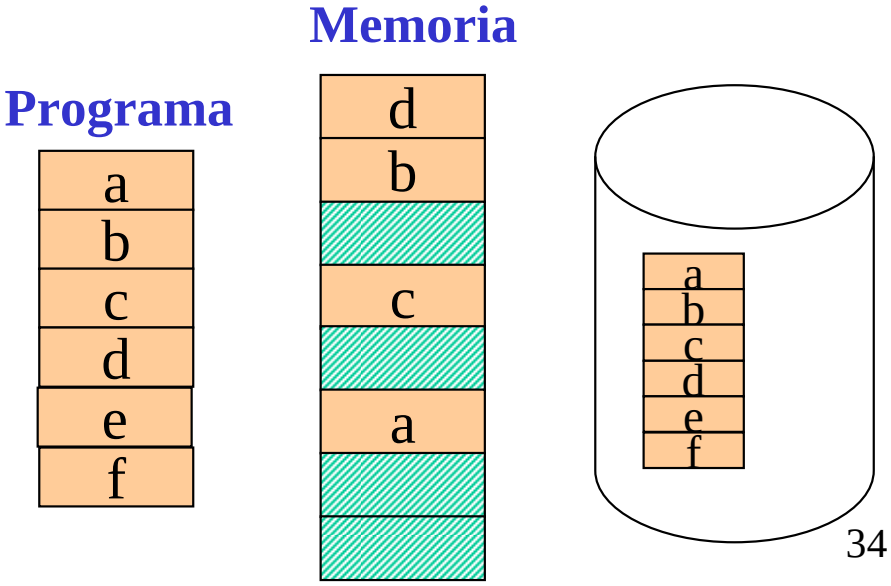



**MMU**
(Memory Management Unit)

# Programa EZ JARRAIAN eta EZ OSORIK

## ❑ EZ OSORIK

- Alegiazko memoria (diskoan, *swap area*)
- Orrikapena edo segmentazioarekin konbinatuz
- Atzipen-hutsegitea (orri-hutsegitea): itzulpena egiteko unean, alegiazko helbide bati dagokion helbide fisikoa ez existitzea
  - Hutsegiteen arreta-errutina: falta den orria diskotik memoriara pasa
  - Ordezkatze-politikak - Hiper-orrikapena (*trashing*): eraginkortasunaren jaitsiera
  - Karga aurreratua / eskaripean



# 5.5 Programen kargarako sistema-deiak

int **execlp**(char * path, char *arg0, char *arg1, ..., char *argn, NULL);

int **execvp**(const char *path, char *const argv[]);

…

// Programa berria kargatzen du, deitzailea ordezkatuz

```
#include <stdlib.h>

int system (const char * agindua);  //   /bin/sh –c agindua
```

**3. Ariketa: jaurti0 probatu, system() eta execlp() erabiliz**

# Tresna interesgarriak

- gcc, as, ld, collect2
- ar : liburutegiak sortzeko
- ldconfig: configure dynamic linker run time bindings
- Exekutagarrien/objektuen/liburutegien azterketa
  - nm : list symbols from object files
  - objdump : display information from object files.
  - file : determine file type

Lotura interesgarriak:

- http://www.faqs.org/docs/Linux-HOWTO/Program-Library-HOWTO.html
- http://www.ibm.com/developerworks/library/l-shobj/