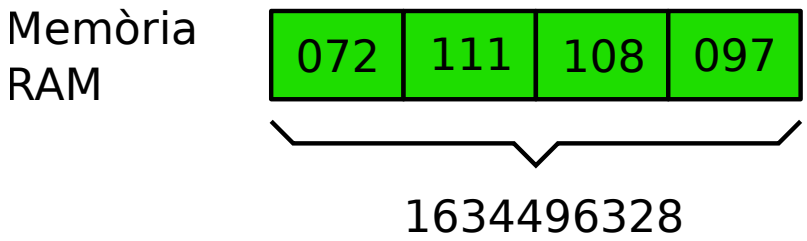
Motivació

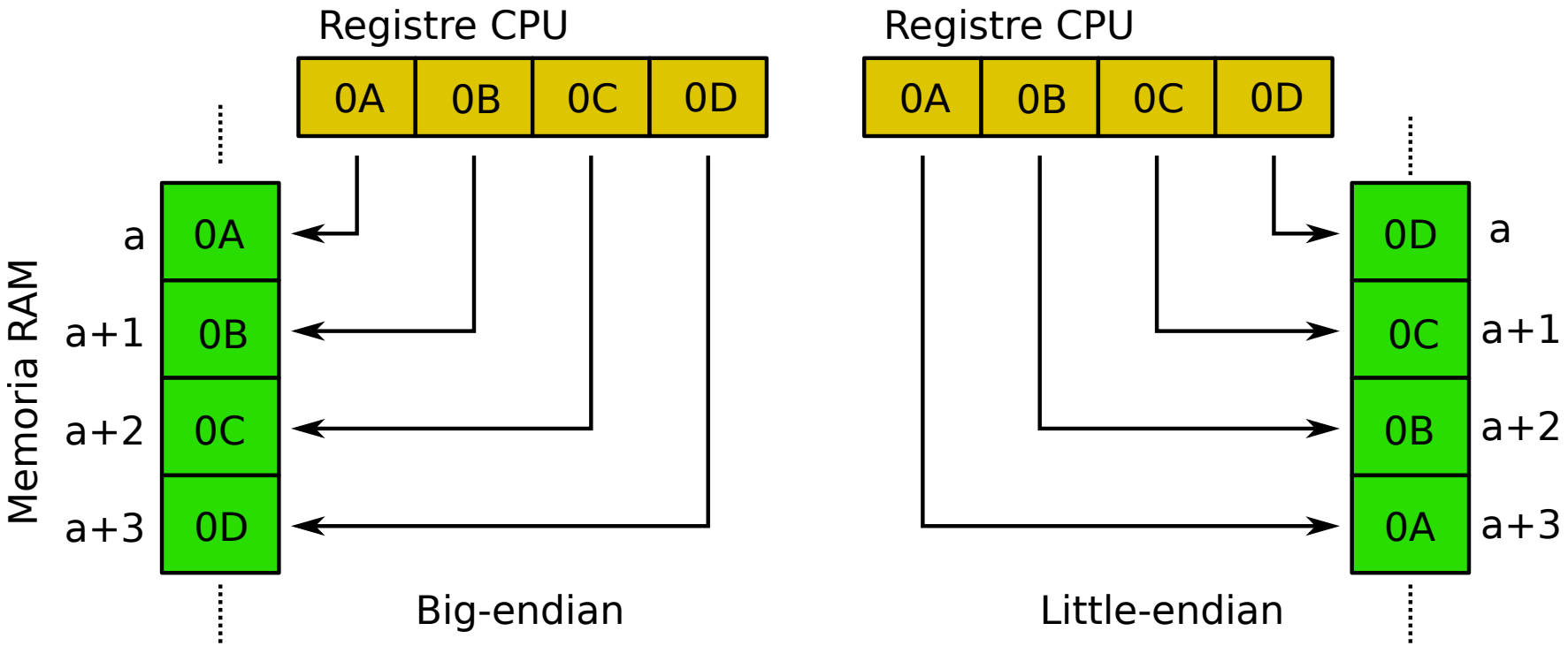
Tornem a l'exemple inicial de les dades no formatades. Suposem que tenim emmagatzemat el següent sencer a memòria RAM.



- En desar el sencer de forma no formatada (`fwrite`) es desen els bytes 072, 111, 108 i 097 a disc.
- Si carreguem aquests bytes (`fread`) en el mateix ordinador en què les hem escrit es llegirà el mateix sencer: 1634496328.
- Es llegirà el mateix sencer si es carrega el fitxer en un ordinador d'una altra arquitectura ?

Endianness

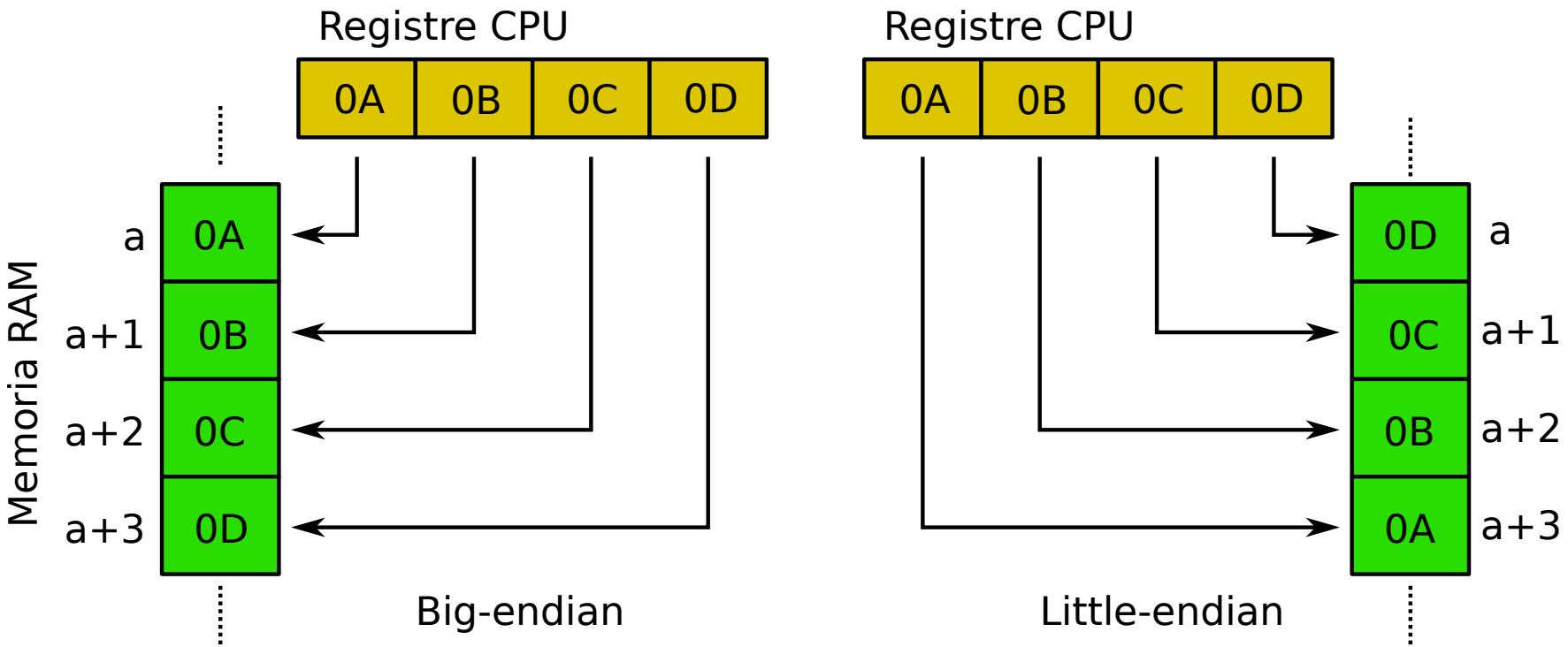
El terme endianness fa referència a la forma en què s'emmagatzemen a memòria (RAM) els tipus bàsics de més d'un byte (sencer, float, double, ...).



Endianness

Al dibuix:

- A la CPU les dades s'emmagatzemen del "Most Significant Byte" (esquerra) al "Least Significant Byte" (dreta).
- Depenent de l'arquitectura, les dades s'emmagatzemen en un ordre diferent a la memòria RAM.



Endianness

Les dues formes més conegudes d'emmagatzemar multi-bytes són:

- Big-endian: el bytes d'un registre s'emmagatzemen a memòria RAM de més a menys significant. És el mètode més natural i s'utilitza per arquitectures IBM, Motorola, ARM, etc.
- Little-endian: el bytes d'un registre s'emmagatzemen a memòria RAM de menys a més significant.. És el mètode utilitzat per arquitectures Intel o AMD.

Com a programadors no ens hem de preocupar quina arquitectura es fa servir llevat que manipulem (per exemple, desem a disc o enviem per xarxa) dades no formatades. Per evitar problemes acostumen a utilitzar-se fitxers formatats (per exemple, XML).

Endianness

Com saber si la CPU en que s'executa el nostre programa és big-endian o little-endian ? Codi `big-little-endian.c`

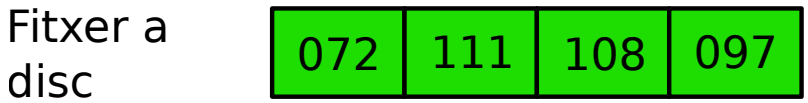
```c
#include <stdio.h>

int main() {
  int a = 0x0A0B0C0D;
  unsigned char *c = (unsigned char *)(&a);

  if (*c == 0x0D)
    printf("little-endian\n");
  else
    printf("big-endian\n");

  return 0;
}
```

Exemple d'execució a un ordinador AMD

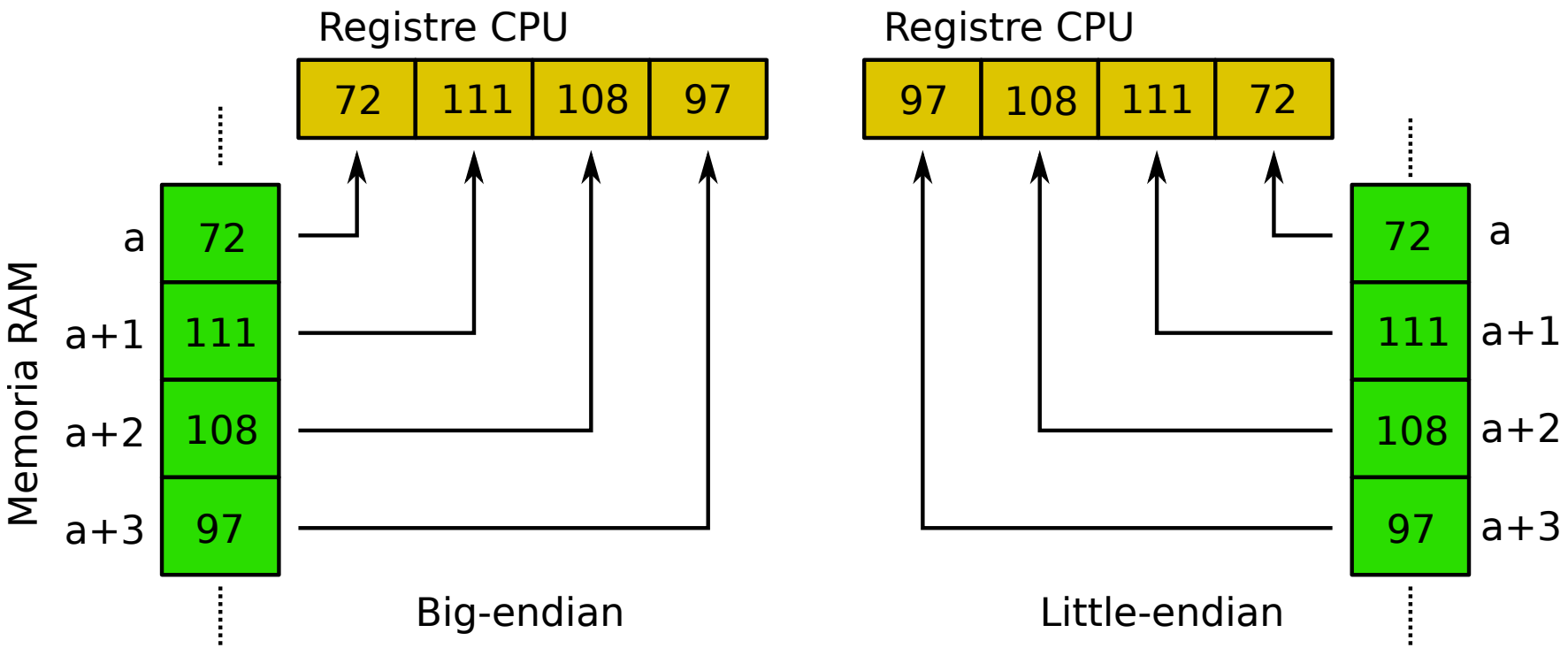
```
$ ./big-little-endian
little-endian
```

Endianness

Suposem que hem guardat aquestes dades associades a un fitxer



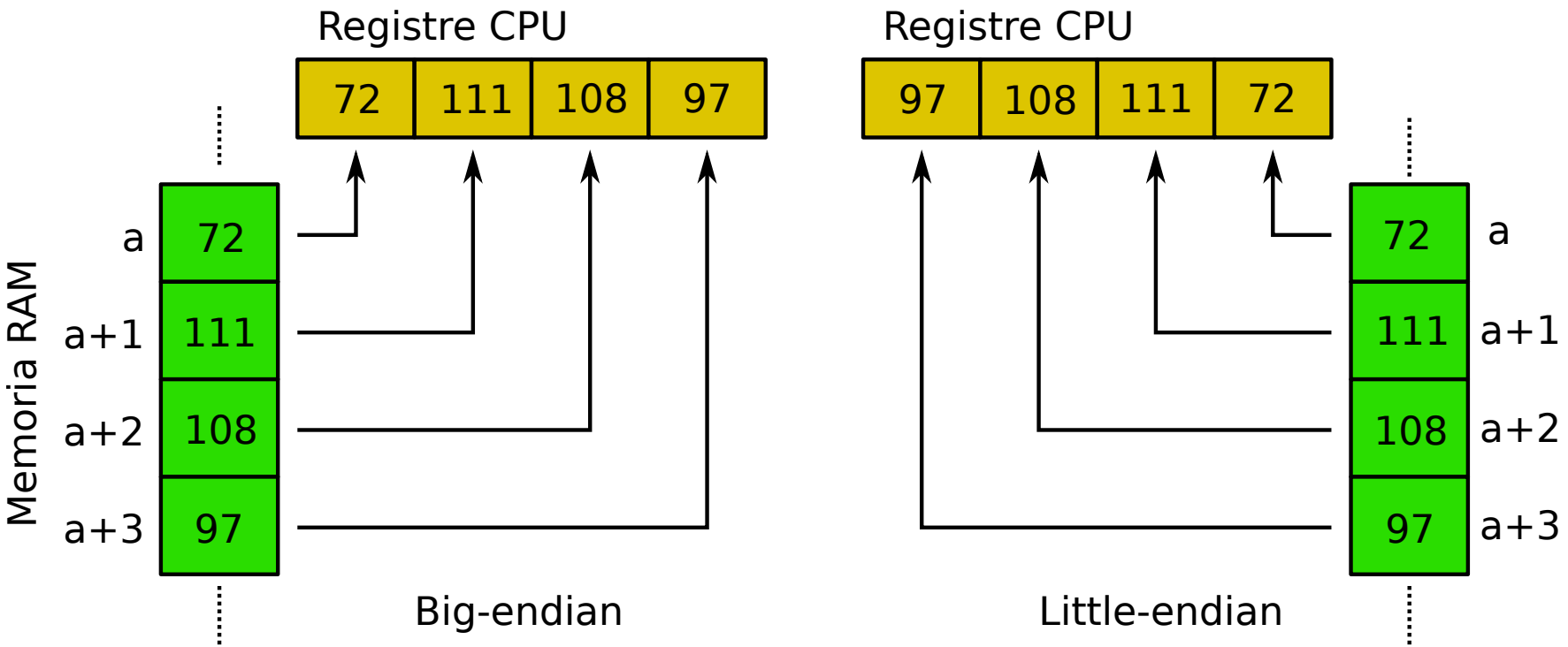
En llegir aquests bytes amb un fread de 4 bytes els guardem a RAM en l'ordre en què apareixen a disc.



Endianness

En operar amb el número el "transferim" a la CPU... i aitx, fatal!

- Amb una CPU big-endian, "veurem"
  $97 + 108 \times 256 + 111 \times 256^2 + 72 \times 256^3 = 1215261793$.
- Amb una CPU little-endian, "veurem"
  $72 + 111 \times 256 + 108 \times 256^2 + 97 \times 256^3 = 1634496328$.



Endianness

Conclusions

- En fer servir fitxers no formatats els valors multi-byte que es llegeixen són diferents depenent de si el programa s'executa en una CPU big-endian o little-endian.
- Per assegurar interoperatibilitat entre diverses arquitectures s'acostumen a desar dades multi-byte en format big-endian.
  - Si l'ordinador és big-endian, es desen les dades tal com estan a memòria.
  - Si l'ordinador és little-endian, cal "intercanviar" els bytes abans de desar-los. En carregar les dades, cal tornar a intercanviar els bytes per interpretar les dades llegides correctament.