Computer Fundamentals

Processor Architecture: Memory Hierarchy

Exercises

Grau en Intel·ligència Artificial

Xavier Martorell

Xavi Verdú

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC)

2021-2022 Q1

Creative Commons License

Exercise 1

- Determine the characteristics of the different architectures, given by their “lstopo” information
  - Processor specs:
    - Total Number of cores
    - Total Number of threads
  - Memory specs:
    - Total RAM Memory
    - L3 memory (shared or private)
    - L2 memory (shared or private)
    - L1D memory (shared or private)
    - L1I memory (shared or private)

Exercise 1.1

Machine (12GB) + Package L#0 + L3 L#0 (6144KB)
  L2 L#0 (512KB) + L1d L#0 (48KB) + L1i L#0 (32KB) + Core L#0
    PU L#0 (P#0)
    PU L#1 (P#1)
  L2 L#1 (512KB) + L1d L#1 (48KB) + L1i L#1 (32KB) + Core L#1
    PU L#2 (P#2)
    PU L#3 (P#3)

Exercise 1.2

Machine (3936MB)
  Package L#0 + L3 L#0 (16MB) + L2 L#0 (4096KB) + L1d L#0 (32KB) + L1i L#0 (32KB) + Core L#0 + PU L#0 (P#0)

Exercise 1.3

Machine (188GB total)
  NUMANode L#0 (P#0 93GB)
    Package L#0 + L3 L#0 (22MB)
      L2 L#0 (1024KB) + L1d L#0 (32KB) + L1i L#0 (32KB) + Core L#0 + PU L#0 (P#0)
      L2 L#1 (1024KB) + L1d L#1 (32KB) + L1i L#1 (32KB) + Core L#1 + PU L#1 (P#1)
      L2 L#2 (1024KB) + L1d L#2 (32KB) + L1i L#2 (32KB) + Core L#2 + PU L#2 (P#2)
      L2 L#3 (1024KB) + L1d L#3 (32KB) + L1i L#3 (32KB) + Core L#3 + PU L#3 (P#3)
  NUMANode L#1 (P#1 94GB)
    Package L#1 + L3 L#1 (22MB)
      L2 L#4 (1024KB) + L1d L#4 (32KB) + L1i L#4 (32KB) + Core L#4 + PU L#4 (P#4)
      L2 L#5 (1024KB) + L1d L#5 (32KB) + L1i L#5 (32KB) + Core L#5 + PU L#5 (P#5)
      L2 L#6 (1024KB) + L1d L#6 (32KB) + L1i L#6 (32KB) + Core L#6 + PU L#6 (P#6)
      L2 L#7 (1024KB) + L1d L#7 (32KB) + L1i L#7 (32KB) + Core L#7 + PU L#7 (P#7)

Exercise 1.4

Machine (94GB total)
  NUMANode L#0 (P#0 47GB)
    Package L#0 + L3 L#0 (33MB)
      L2 L#0 (1024KB) + L1d L#0 (32KB) + L1i L#0 (32KB) + Core L#0 + PU L#0 (P#0)
      L2 L#1 (1024KB) + L1d L#1 (32KB) + L1i L#1 (32KB) + Core L#1 + PU L#1 (P#1)
... ...
      L2 L#23 (1024KB) + L1d L#23 (32KB) + L1i L#23 (32KB) + Core L#23 + PU L#23 (P#23)
  NUMANode L#1 (P#1 47GB) + Package L#1 + L3 L#1 (33MB)
    L2 L#24 (1024KB) + L1d L#24 (32KB) + L1i L#24 (32KB) + Core L#24 + PU L#24 (P#24)
    L2 L#25 (1024KB) + L1d L#25 (32KB) + L1i L#25 (32KB) + Core L#25 + PU L#25 (P#25)
... ...
    L2 L#47 (1024KB) + L1d L#47 (32KB) + L1i L#47 (32KB) + Core L#47 + PU L#47 (P#47)

Exercise 1.5

Machine (126GB total) + Package L#0
NUMANode L#0 (P#0 31GB)
L3 L#0 (8192KB)
L2 L#0 (512KB) + L1d L#0 (32KB) + L1i L#0 (64KB) + Core L#0
PU L#0 (P#0)
PU L#1 (P#24)
L2 L#1 (512KB) + L1d L#1 (32KB) + L1i L#1 (64KB) + Core L#1
PU L#2 (P#1)
PU L#3 (P#25)
L2 L#2 (512KB) + L1d L#2 (32KB) + L1i L#2 (64KB) + Core L#2
PU L#4 (P#2)
PU L#5 (P#26)
L3 L#1 (8192KB)
L2 L#3 (512KB) + L1d L#3 (32KB) + L1i L#3 (64KB) + Core L#3
PU L#6 (P#3)
PU L#7 (P#27)
L2 L#4 (512KB) + L1d L#4 (32KB) + L1i L#4 (64KB) + Core L#4
PU L#8 (P#4)
PU L#9 (P#28)
L2 L#5 (512KB) + L1d L#5 (32KB) + L1i L#5 (64KB) + Core L#5
PU L#10 (P#5)
PU L#11 (P#29)

HostBridge L#0
PCIBridge
PCIBridge
PCI 1a03:2000

NUMANode L#1 (P#1 31GB)
L3 L#2 (8192KB)
L2 L#6 (512KB) + L1d L#6 (32KB) + L1i L#6 (64KB) + Core L#6
PU L#12 (P#6)
PU L#13 (P#30)
L2 L#7 (512KB) + L1d L#7 (32KB) + L1i L#7 (64KB) + Core L#7
PU L#14 (P#7)
PU L#15 (P#31)
L2 L#8 (512KB) + L1d L#8 (32KB) + L1i L#8 (64KB) + Core L#8
PU L#16 (P#8)
PU L#17 (P#32)
L3 L#3 (8192KB)
L2 L#9 (512KB) + L1d L#9 (32KB) + L1i L#9 (64KB) + Core L#9
PU L#18 (P#9)
PU L#19 (P#33)
L2 L#10 (512KB) + L1d L#10 (32KB) + L1i L#10 (64KB) + Core L#10
PU L#20 (P#10)
PU L#21 (P#34)
L2 L#11 (512KB) + L1d L#11 (32KB) + L1i L#11 (64KB) + Core L#11
PU L#22 (P#11)
PU L#23 (P#35)

NUMANode L#2 (P#2 31GB)
L3 L#4 (8192KB)
L2 L#12 (512KB) + L1d L#12 (32KB) + L1i L#12 (64KB) + Core L#12
...
L2 L#13 (512KB) + L1d L#13 (32KB) + L1i L#13 (64KB) + Core L#13
...
L2 L#14 (512KB) + L1d L#14 (32KB) + L1i L#14 (64KB) + Core L#14
L3 L#5 (8192KB)
L2 L#15 (512KB) + L1d L#15 (32KB) + L1i L#15 (64KB) + Core L#15
L2 L#16 (512KB) + L1d L#16 (32KB) + L1i L#16 (64KB) + Core L#16
L2 L#17 (512KB) + L1d L#17 (32KB) + L1i L#17 (64KB) + Core L#17

NUMANode L#3 (P#3 31GB)
L3 L#6 (8192KB)
L2 L#18 (512KB) + L1d L#18 (32KB) + L1i L#18 (64KB) + Core L#18
...
L2 L#19 (512KB) + L1d L#19 (32KB) + L1i L#19 (64KB) + Core L#19
...
L2 L#20 (512KB) + L1d L#20 (32KB) + L1i L#20 (64KB) + Core L#20
...
L3 L#7 (8192KB)
L2 L#21 (512KB) + L1d L#21 (32KB) + L1i L#21 (64KB) + Core L#21
...
L2 L#22 (512KB) + L1d L#22 (32KB) + L1i L#22 (64KB) + Core L#22
...
L2 L#23 (512KB) + L1d L#23 (32KB) + L1i L#23 (64KB) + Core L#23
PU L#46 (P#23)
PU L#47 (P#47)
HostBridge L#8
PCIBridge
PCI 1022:7901
Block(Disk) L#2 "sda"

Exercise 2

- Determine how the program accesses RAM memory and the architecture brings the data into the L1 data cache

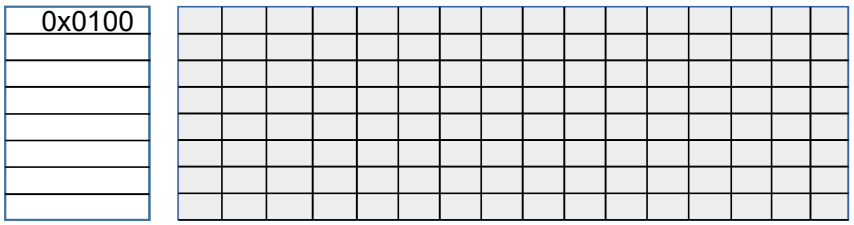
Cache
8 lines
16 bytes per line
total: 128 bytes

Assume
&v[0] == 0x0100

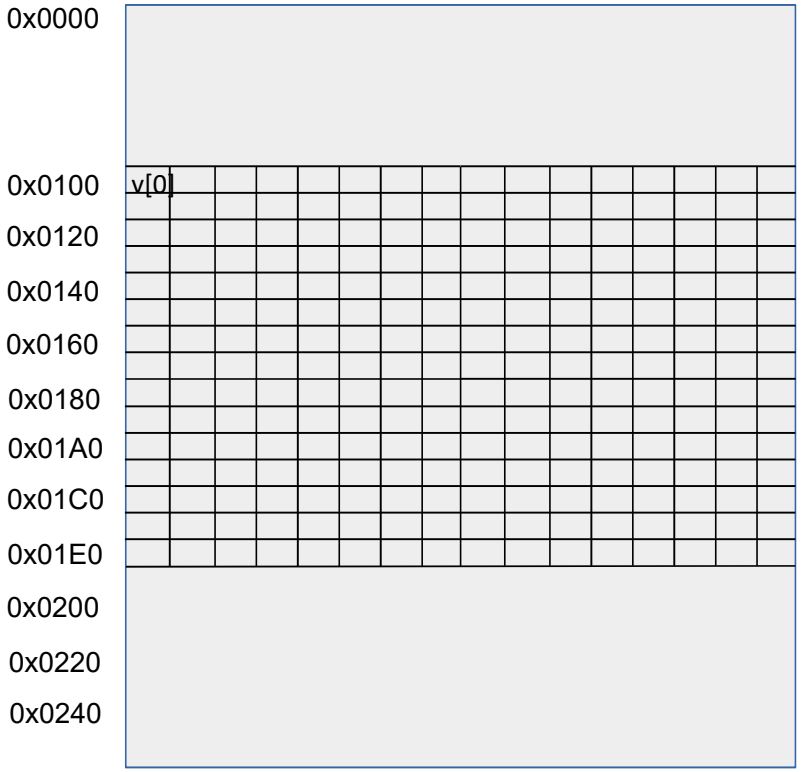
Program

```
int v[80];
int r =0;
int i;
for (i=0; i < 80; i++) {
    r += v[i];
}
```

L1 data cache



RAM



Exercise 3

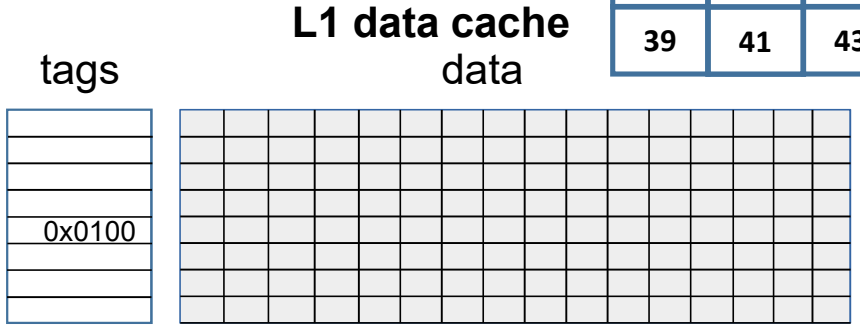
- Determine how the “matrix” variable is laid out on the RAM memory using **row-major** mapping and **little-endian**

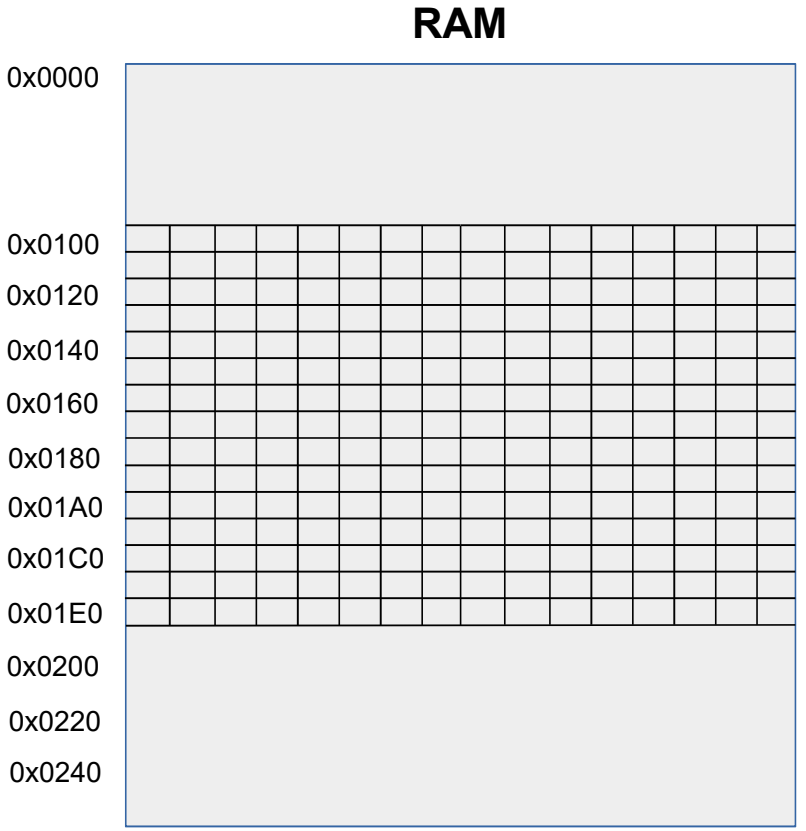
Cache
8 lines
16 bytes per line
total: 128 bytes

Assume
&matrix[0][0] == 0x0100

Data
int matrix [6][6];

4	6	8	10	12	14
13	15	17	19	21	23
22	24	26	28	30	32
31	33	35	37	39	41
40	42	44	46	48	50
39	41	43	45	47	49





Exercise 4

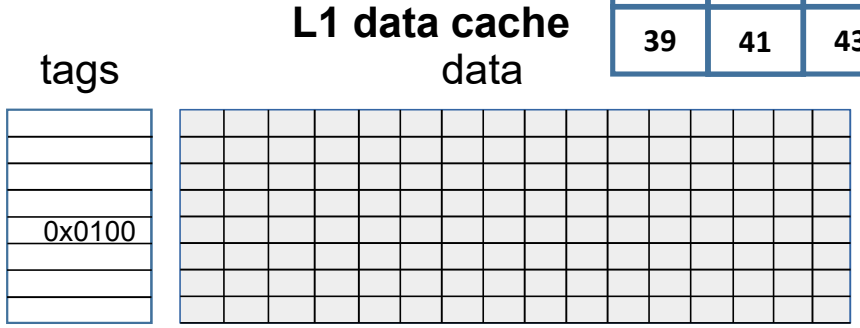
- Determine how the “matrix” variable is laid out on the RAM memory using **column-major** mapping and **little-endian**

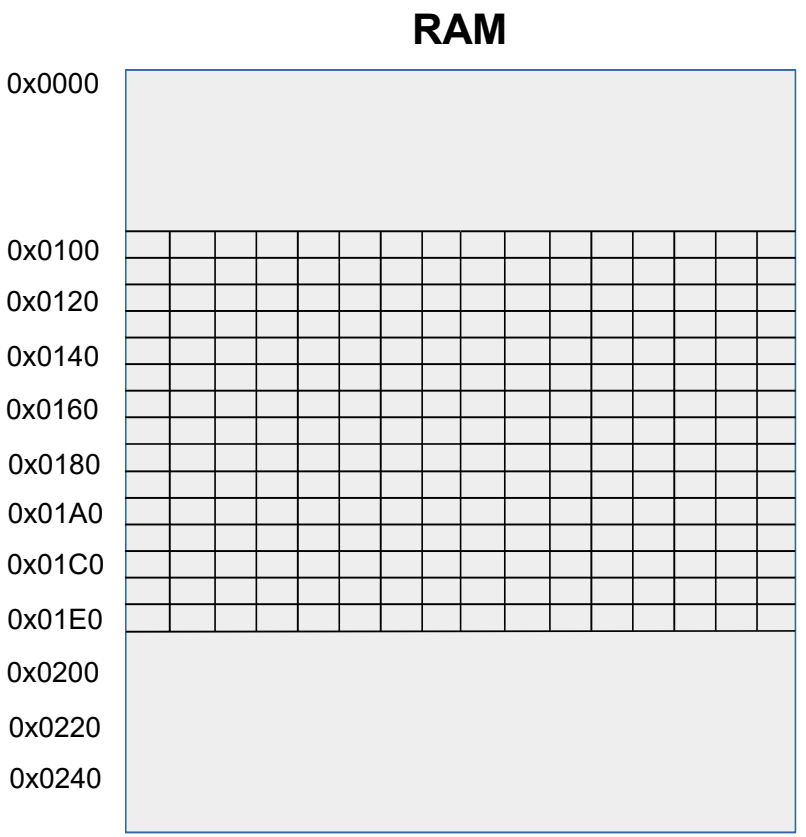
Cache
8 lines
16 bytes per line
total: 128 bytes

Assume
&matrix[0][0] == 0x0100

Data
int matrix [6][6];

4	6	8	10	12	14
13	15	17	19	21	23
22	24	26	28	30	32
31	33	35	37	39	41
40	42	44	46	48	50
39	41	43	45	47	49





Exercise 5

- Indicate “Hit”, “Miss” or “---” according to the accesses done to the mem hierarchy

Cache hierarchy
L1-Ins.: 8 lines; 64B/line
L1-Data: 8 lines; 64B/line
L2: 16 lines; 256B/line

- **Do you identify any form of locality?**

Order	Instr/ data	@	L1 Ins.	L1 Data	L2 (Ins. & Data)
1	Instr.	0x001000			
2	Instr.	0x001004			
3	Dada	0x00B000			
4	Dada	0x00B004			
5	Instr.	0x001000			
6	Instr.	0x001004			
7	Dada	0x00B008			
8	Dada	0x00B00C			
9	Instr.	0x001000			
10	Instr.	0x001004			
11	Dada	0x00B010			
12	Dada	0x00B014			

Exercise 6

- Indicate “Hit”, “Miss” or “---” according to the accesses done to the mem hierarchy

Cache hierarchy
L1-Ins.: 8 lines; 64B/line
L1-Data: 8 lines; 64B/line
L2: 16 lines; 256B/line

- **Do you identify any form of locality?**
- **Compare locality with Exercise 5.**

Order	Instr/ data	@	L1 Ins.	L1 Data	L2 (Ins. & Data)
1	Instr.	0x002000			
2	Instr.	0x002004			
3	Dada	0x010000			
4	Dada	0x010084			
5	Instr.	0x002008			
6	Instr.	0x00200C			
7	Dada	0x010108			
8	Dada	0x010118			
9	Instr.	0x002200			
10	Instr.	0x002204			
11	Dada	0x0100F0			
12	Dada	0x010200			