Subject : Date :

<< memory >>

- يعني إيه CA ؟
مكونات الحاسب ، تركيب الأوامر ، نقل الأوامر ، تنظيم الذاكرة
تمثيل الأرقام ، العمليات الحسابية والمنطقية .

* memory :-
>> RAM : Random - Access - Memory .
- عبارة عن ذاكرة تخزين مؤقتة تنفقد الى فيها بمجرد فصل التيار عنها
وهي عبارة SRAM , DRAM
- DRAM :
- عبارة عن مكثفات تسرب الطاقة ببطء لذا
تحتاج إلى عملية refresh مستمرة لمنع فقد البيانات .
* - SRAM :
عبارة عن دوائر متشابهة لـ D-flipflop وهي ذاكرة سريعة لا تحتاج إلى refresh

>> ROM : Read-only- Memory .
- عبارة عن ذاكرة دائمة لا تفقد البيانات بفصل البيانات عنها لذلك تستخدم بشكل
أكبر في تخزين البرامج .

* memory hierarchy :
- كل ما تكون سريعة وصغيرة تحفظ في الـ CPU .
- ~ ~ ~ أكبر وأبطأ تبعد عن الـ CPU لذلك تحتاج إلى database .
→ register → Cache → main memory → Virtual
. register : ذاكرة صغيرة مؤقتة موجودة في الـ CPU .
. virtual : عبارة عن الـ hard .
- لو عاوزين نـ access بيانات معينة
الـ CPU هتبعت request لأقرب memory (cache) لو مش موجودة هتروح
لـ main ، لو مش موجودة هتروح لـ disk لو لقتها بتيجي بالـ Cache .

Subject : Date :

- hit → الداتا اللي عاوزاها موجودة في أي level.
- miss → الداتا مش موجودة.
- hit rate → الوقت اللي لقيت فيه الداتا.
- miss rate → الوقت اللي ملقتش فيه الداتا = 1-hit rate

* Principle of locality :-

- بمجرد ما عمل access لداتا معينة. معناه إني هحتاج الداتا اللي جنبها وليها أنواع -

(1) Temporal locality :- آخر داتا عملت عليها access هحتاجها تاني

(2) spatial locality :- تميل إلى تجميع الداتا.

(3) sequential locality :- هعمل access للداتا ورا بعض.

* الغرض من الـ Cache إني أسرع عملية الـ access للداتا عن طريق إني حطيتها في مكان قريب من الـ CPU.

[Direct Cache mapping]

- الـ main memory عبارة عن مجموعة من الـ blocks وكل block عبارة عن عدد من الـ words وعند الـ mapping يتم نسخ الـ block كامل لـ line معين في الـ Cache ولازم حجم الـ line يساوي حجم الـ words في الـ block.

- اسمها direct لأن كل block في الـ main عارف هو رايح فين في الـ Cache
مثلا لو عندي m من الـ blocks في الـ main

Cache : 0 → 0, m, 2m
1 → 1, m+1, 2m+1, ...

* الـ main memory address بيتقسم لمجموعة من الـ fields:

Tag	Block	offset

offset :- بنختار من خلاله أي word داخل ال block الواحد.
Block :- بنختار بيه ال line داخل ال Cache.
Tag :- باقي ال bits ع ال address.

- عشان أعرف أحل المسائل محتاج إيه؟!
- أعداد ال main memory address bits.
- عدد ال words داخل ال block.

ex:- Consider a byte-addressable main memory consist of 4 blocks and Cache with 2 blocks where each block is 4 byte.

∴ main memory size = 4 * 4 = 16 byte.
main memory address bits = 4 bits.
offset = 2 bit
block = 1 bit
tag = 1 bit

1	1	2
Tag	block	offset

- main memory format -

ex:- Assume a byte-addressable memory consist of 2^{14} byte, Cache has 16 blocks, and each block has 8 bytes.

- memory size = 2^{14} byte.
- address bits = 14 bits.
offset = 3 bits.
blocks = 4 bits.
tag = 7 bits.

Subject : _____ Date : _____

ex: Assume a byte addressable memory consist of 16 byte divided into 8 blocks, cache contain 4 block.
- memory address = 4 bit.
- number of words in each block: 2 words.
offset = 1 bit. Block = 2 bit tag = 1 bit.

ex: Consider 16-bit memory address and 64 block of cache where each block contain 8 bytes.
memory address = 16 bit.
memory size = 2^{16}
offset = 3 bit, block = 6 bit, tag = 7 bits.

* عرفت إزاي القيمة داخل الـ Cache؟ (إزاي بعمل search فيها).
- بعت address معين للـ Cache، بشوف جزء الـ line المطابق لهذا الـ address بشوف الـ Tag عشان يشوف الـ block كمان موجود ولا لا.
- [علة المقارنة سريعة بس دي مش أحسن طريقة لأن لو عندي مكان فاضي في الـ Cache مش بقدر أضيف فيه عشان أماكن معينة].

* associative mapping *
* الـ block مش ليه مكان معين، عكس أدخل في أي مكان في الـ Cache.

Tag	offset

- لما يجيلي address معين أشوف هذي كام word داخل الـ block وباقي الـ address هيروح للـ Tag.

- العيب: عشان أعمل مقارنة لازم أمشي على كل الـ lines.

Subject : ____________ Date : ____________

* **Set associative :-** [جمعنا بين مميزات الـ associative direct]

في هذه الطريقة تم تقسيم الـ lines لمجموعات من الـ sets

[كدا زودت الـ line اللي ممكن أخزن فيها وقللت عدد الـ lines اللي هدور فيها]:

Tag	set	word

* عند المقارنة هروح مباشرة داخل الـ set واشوف بعدها الـ Tag لأقارن.

ex: two way associative has lines of 16 bytes and the total size 8K byte, 64 M byte of main memory that is byte addressable.

Two-way associative → each set has 2 block.

Total Cache size = 8K = $2^3 * 2^{10} = 2^{13}$ byte.

memory size = $2^6 * 2^{20} = 2^{26}$ byte.
address bits.

number of lines = $\frac{2^{13}}{2^4} = 2^9$

number of set = $\frac{2^9}{2^1} = 2^8$ → bit

tag	set	offset
14	8	4

* special Cases :-
- if K = 1 Then, K-way mapping become direct mapping.
- if K = total number of lines - - - fully-associative mapping.

Subject : _____ Date : _____

ex :- we are using 2-way set mapping with byte addressable main memory of 2^{14} byte and a cache with 16 block, each block contain 8 byte :

number of sets = 8 sets.

tag	set	offset
8 bit	3 bit	3 bit

ex :- suppose a byte-addressable memory contain 2^{20} → address. 1 MB and cache consist of 32 line where each block contain 16 byte using (1) direct. (2) Full associative. (3) 4-way set associative determine where the main memory address 0X326A0 map in cache?

- direct

tag	block	offset
11 bit	5 bit	4 byte

- fully-associative

tag	offset
16 bit	4 byte

- 4-way

tag	set	offset
13 bit	3 bit	4 bit