7

14

1680 sp

نظري

كلية الهندسة المعلوماتية

السنة الثالثة

Memory Hierarchy &Cache memory

د. خولة العلي

RB Informatics 18/11/2024

بنيان الحواسيب 2

قلنا سابقاً أن مصممي الحاسوب يسعون للوصول لجهاز يملك أفضل أداء من جميع النواحي، و من المؤكد أن الذاكرة كانت إحدى أهم الأجزاء التي يسعى المصممون لتطويرها لتقدم أداء أفضل من حيث سرعة الوصول و السعة و تخفيض الكلفة، لذلك تم اللجوء لفكرة هرمية الذواكر (مستويات الذاكرة).

مبدأ التقارب principle of locality

ينتج هذا المبدأ للبرنامج الحالي الوصول إلى جزء صغير من العناوين في أي وقت،ويقسم إلى نوعين:

1. التقارب الزمني (Temporal locality):

التعليمات التي تم الوصول لها مؤخراً يمكن أن يُعاد استخدامها مرة أخرى في وقت قريب.

مثل الحلقات.

Address	
0	Addi: $t1, $zero, 10
4	Loop: add $s0, $s1, $s2
8	Addi: $s2, $s2, 5
12	Addi: $t1, $zero, -1
16	Bne: $t1, $zero, Loop

2. التقارب المكاني (Spatial locality):

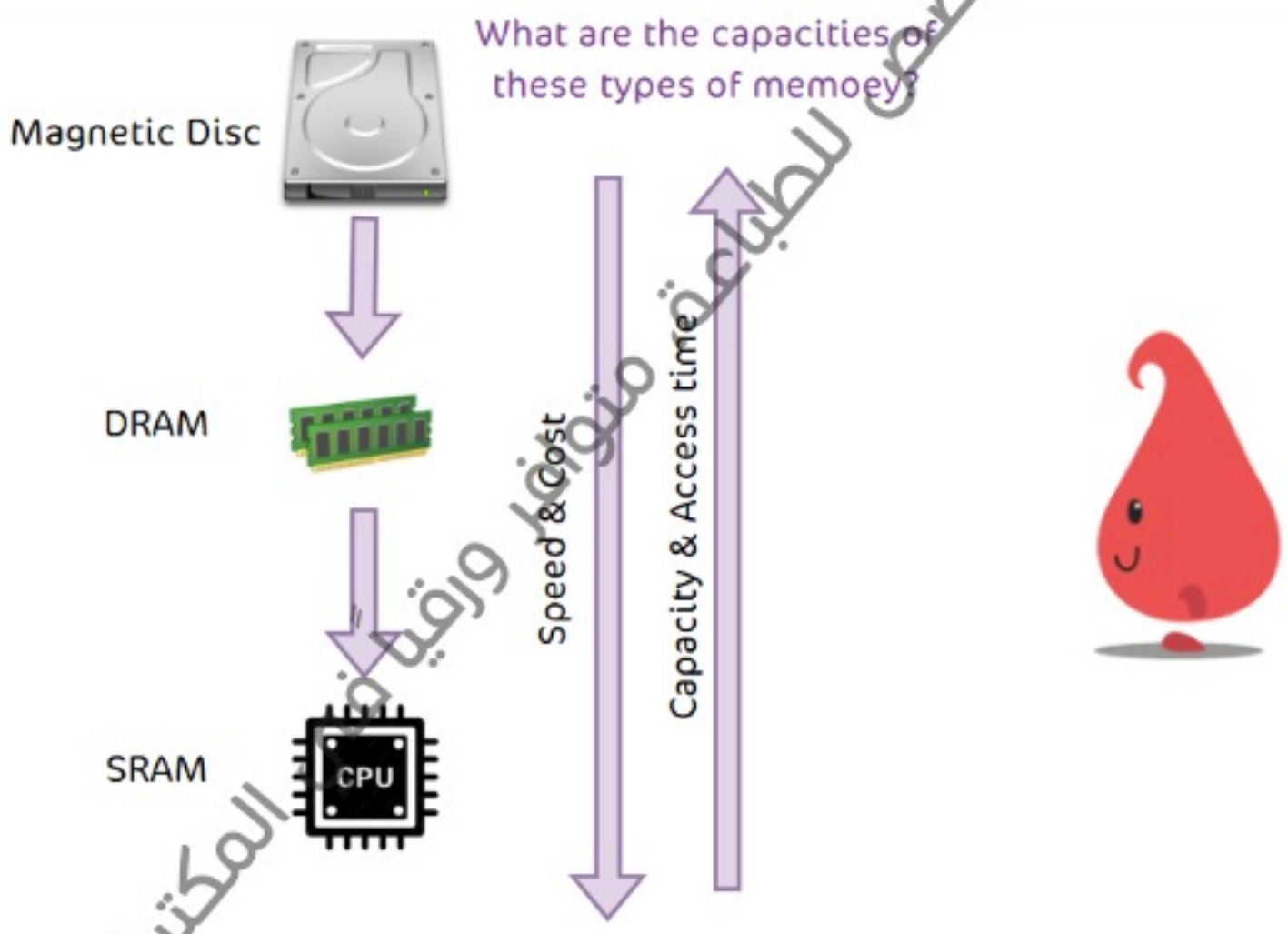
التعليمات القريبة من الشيء الذي تم الوصول إليه مؤخراً في الذاكرة يمكن استخدمها في وقت قريب.

مثل التعليمات المتسلسلة و مصفوفة البيانات.

Address	
40	A[5]
44	A[6]
48	A[7]
52	A[8]

- تعليمات مخزنة في الرام، في كل مرور على الحلقة نحتاج للوصول لذاكرة RAM لنجلب منها التعليمات، لذلك نأتي بالتعليمات اللازمة على شكل Block و يتم وضعها في ذاكرة الـ cache التي تقع ضمن وحدة المعالجة المركزية.
- حسب هرمية الذواكر فإن زمن الوصول لذاكرة الـ cache أقل من زمن الوصول للـ RAM لأنها داخل الـ cpu، و حجمها أصغر من الـ RAM.
- و في مثال المصفوفة فمثلاً عندما نسلسل عناصر المصفوفة ضمن برنامج ما نحتاج للعناصر واحد تلو الآخر، لذلك نأتي بـ Block المصفوفة و نضعه في الـ cache.

Advantages of locality



في المخطط سابقاً، إن الـ magnetic disc في أسفل الهرم، يليه DRAM، يليها SRAM.

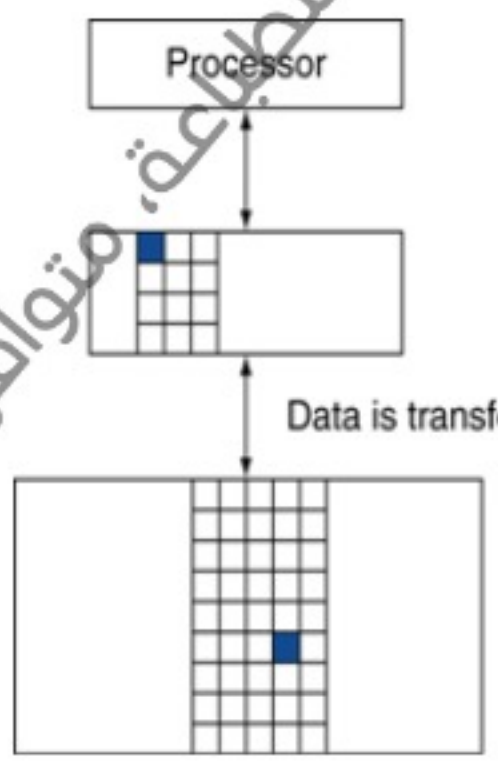
- يتم تخزين أي شيء في القرص (disc) فهو ذاكرة دائمة.
- يتم نسخ معلومات تم طلبها من المعالج و المعلومات المجاورة لها من الـ disc إلى ذاكرة DRAM (Dynamic Random Access memory) و هي أصغر من disc.
  مثل: main memory و هي عبارة عن ترانزستورات.
- ثم تنسخ من DRAM إلى SRAM (Static Random Access memory)
  مثل: cache memory attached to cpu و هي عبارة عن قلابات (flip-flops).

Memory Hierarchy Levels

إن السجلات و الـ Buffers ذواكر مؤقتة صغيرة الحجم و الوصول إليها سريع حيث أنها توجد داخل المعالج و المتحكمات، و كلفتها عالية.

ذاكرة الـ cache مكونة من مستويات، منها ما يوجد داخل الـ cpu و منها ما هو خارجها و مجاور لها، طبعاً إن الوصول للمستويات الداخلية أسرع.

Block: وحدة نقل معلومات (عند نقل جزء من المعلومات من ذاكرة لأخرى يسمى هذا الجزء Block).
يمكن أن تكون عدة كلمات، بالشكل النموذجي في معالج (Intel core i7) حجمه 64bytes = 512bit.



Data is copied between only two adjacent levels at a time.

إذا كانت الداتا التي يتم الوصول إليها موجودة في المستويات العالية (في الكاش مثلاً) يسمى ذلك Hit و هو الوصول للداتا من المستوى الأعلى.

إذا لم توجد الداتا في الـ cache، يسمى ذلك Miss حيث لم نجد المعلومة فيتم نسخها من الطبقة التي قبلها (في الأسفل).

$$Hit\ ratio = \frac{hits}{accesses}$$

$$Miss\ ratio = \frac{misses}{access} = 1 - Hit\ ratio$$

Hit time: الوقت اللازم للوصول للمستوى الأعلى.

Miss penalty: الوقت اللازم لنسخ الـ Block من مستوى أدنى لمستوى أعلى و تبديله بالقديم، مضافاً إليه الوقت المستغرق لتسليم المعلومة للمعالج.

و بعدها تصبح التعليمات في المستوى الأعلى و يتم الوصول إليها.

ملاحظة:

المعلومات التي يتم تبادلها لا تنتقل و إنما تنسخ، أي يبقى هناك نسخة منها في المستويات الأدنى، ولا يتم تبادل المعلومات إلا بين طبقتين (أو مستويين) متتاليين.

k, k-1 أو k+1 , k

مثال:

لنفترض هناك برنامج يحتاج 2000 تعليمة (تحميل و تخزين)، منها 1250 تعليمة موجودة في ذاكرة الـ cache، و 750 تعليمة يتم تزويد المعالج بها من خلال الـ main memory أو disc memory.

احسب: Miss rate و Hit rate لذاكرة الـ cache

الحل:

$$Miss\ rate = \frac{misses}{accesses} = \frac{750}{2000} = 0.375 = 37.5\%$$ نسبة

(هذا الجواب الذي نريده)

$$Hit\ rate = \frac{Hits}{acceses} = 1 - miss\ rate$$

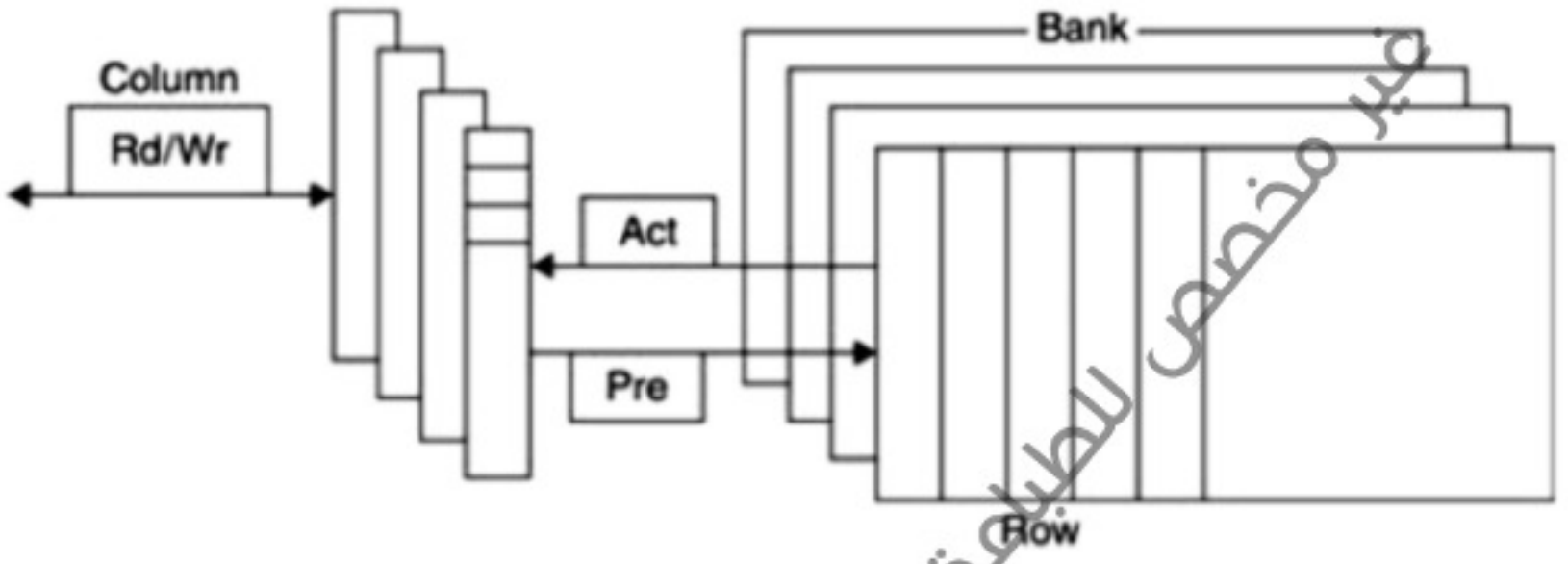
$$= \frac{1250}{2000} = 1 - 0.375 = 0.625 = 62.5\%$$

Memory technology

	زمن الوصول	الكلفة
Static RAM (SRAM)	0.5 → 2 (ns)	2000 → 5000 $/GB
Dynamic RAM (DRAM)	50 → 70 (ns)	20 → 75 $/GB
Magnetic disc	5 → 20 (ms)	0.20→ 2 $/GB
Ideal memory	زمن وصول ذاكرة الـ SRAM	كلفة و سعة الـ disc

DRAM Technology

- يتم تخزين البيانات على شكل شحن للمكثفة.
- كل ترانزستور يصل إلى البيانات من أجل القراءة أو الكتابة.
- لذلك يجب تجديدها (refresh) بشكل دوري.

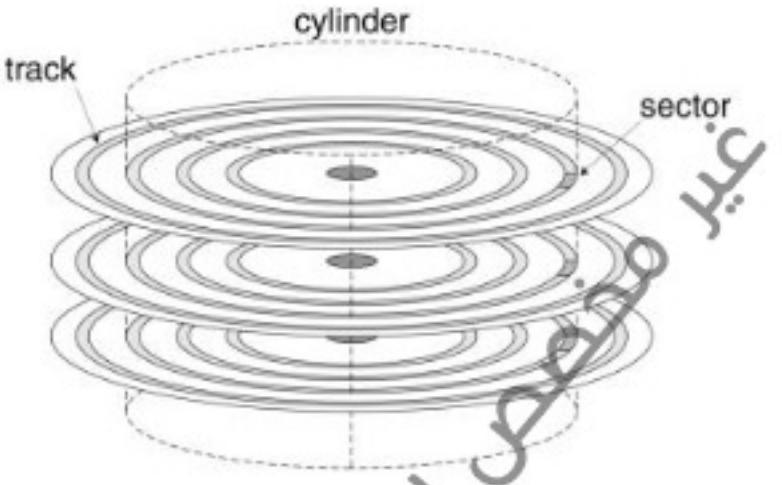


- ♥ عند قراءة بيانات أو كتابتها، فإنه يتم الحفاظ على هذه البيانات ضمن النواقل لعدة أجزاء من الثانية.
- ♥ عملية الـ refresh تتم في سطر من الـ DRAM.
- ♥ يتم تنظيم السطور ضمن Banks حيث كل منها يتضمن سلسلة من السطور.
- ♥ precharge (pre) لفتح أو إغلاق Bank محدد.
- ♥ activate (Act) تسبب نقل السطر إلى ذاكرة الـ Buffer.

Flash storage:

- من أشباه الموصلات التي تستخدم للتخزين و هي ذاكرة غير متطايرة (غير مؤقتة).
- أسرع من القرص بـ 100 ← 1000 مرة.
- أكثر متانة و أقل استهلاك للطاقة و حجمها أصغر.
- كلفتها تتراوح بين القرص و ذاكرة DRAM.

Disk storage:



- تتألف من مجموعة اسطوانات متحدة المركز و متحركة.
- لقراءة معلومات من القرص الصلب أو الكتابة عليه، هنالك ذراع متحرك يحوي على مجمّع كهرومغناطيسي يسمى رأس القراءة و الكتابة يقع فوق كل سطح لاسطوانة.
- سطح كل قرص يقسم إلى مسارات (أكبر من 10000 مسار).
- كل مسار يقسم إلى قطاعات تحتوي على بيانات.

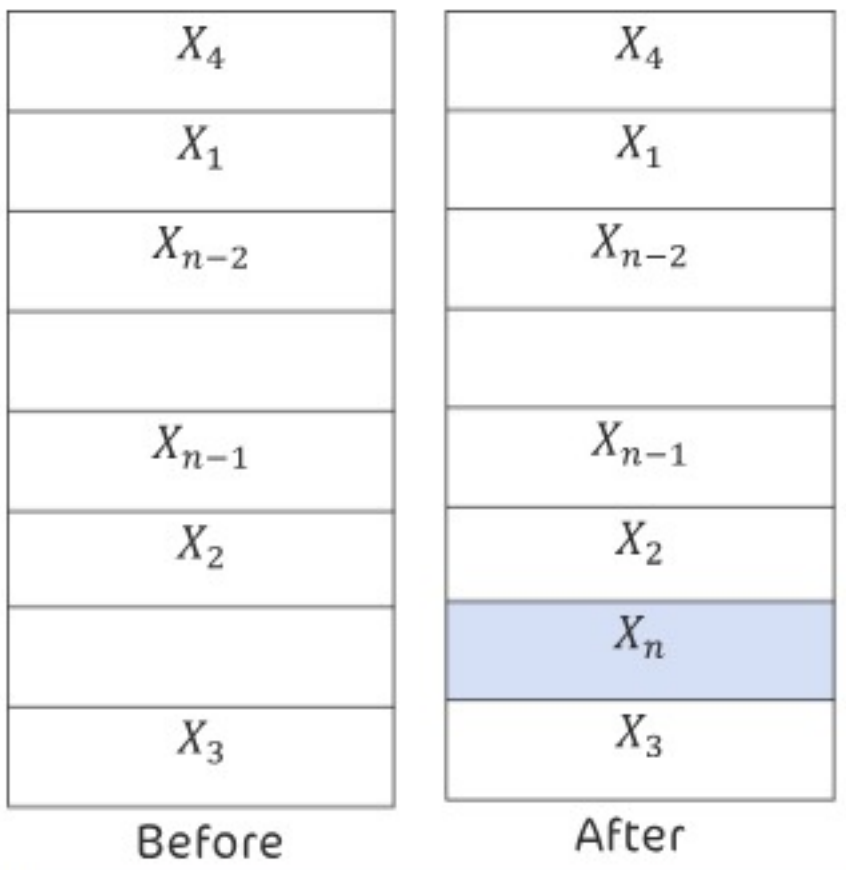
كل قطاع يسجل:

- رقم تعريف القطاع (ID)
- بيانات تتراوح بين 512 → 4096 بايت
- كود تصحيح اخطاء (ECC) Error Correction Code
- مجالات التزامن

Keep going

Cache Memory

- ذاكرة صغيرة سريعة الوصول حيث تمثل مثل ذاكرة الـ Buffer من حيث السرعة و المساحة.
- و تعد الأقرب من بين الذواكر بالنسبة للـ cpu.
- تحمل مجموعة التعليمات ذات الاستخدام المتكرر.



طلب المعالج الكلمة X_n و هي غير موجودة في ذاكرة الـ cache.

سينتج حالة Miss.

يتم جلب X_n من الذاكرة و وضعها في الـ cache.

Cache Organization

Mapping: هي العلاقة بين عنوان البيانات في الذاكرة الرئيسية و موقعها في الذاكرة المؤقتة (cache).

- كل عنوان ذاكرة يتوضع في مجموعة واحدة معينة من الـ cache.
- تم تقسيم الـ cache بناء على عدد الـ Blocks في المجموعة و تم ذلك في كل نوع كالآتي:

In a:

- Fully associative cache
- N-way set associative cache
- Direct mapped cache

1. Direct mapped cache:

1) كل مجموعة تحوي تماماً Block واحد فقط.
2) عدد المجموعات = عدد الـ Blocks.
3) كل عنوان ذاكرة يتوضع في مكان محدد من الـ Block في الذاكرة المؤقتة.

2. N-way set associative cache:

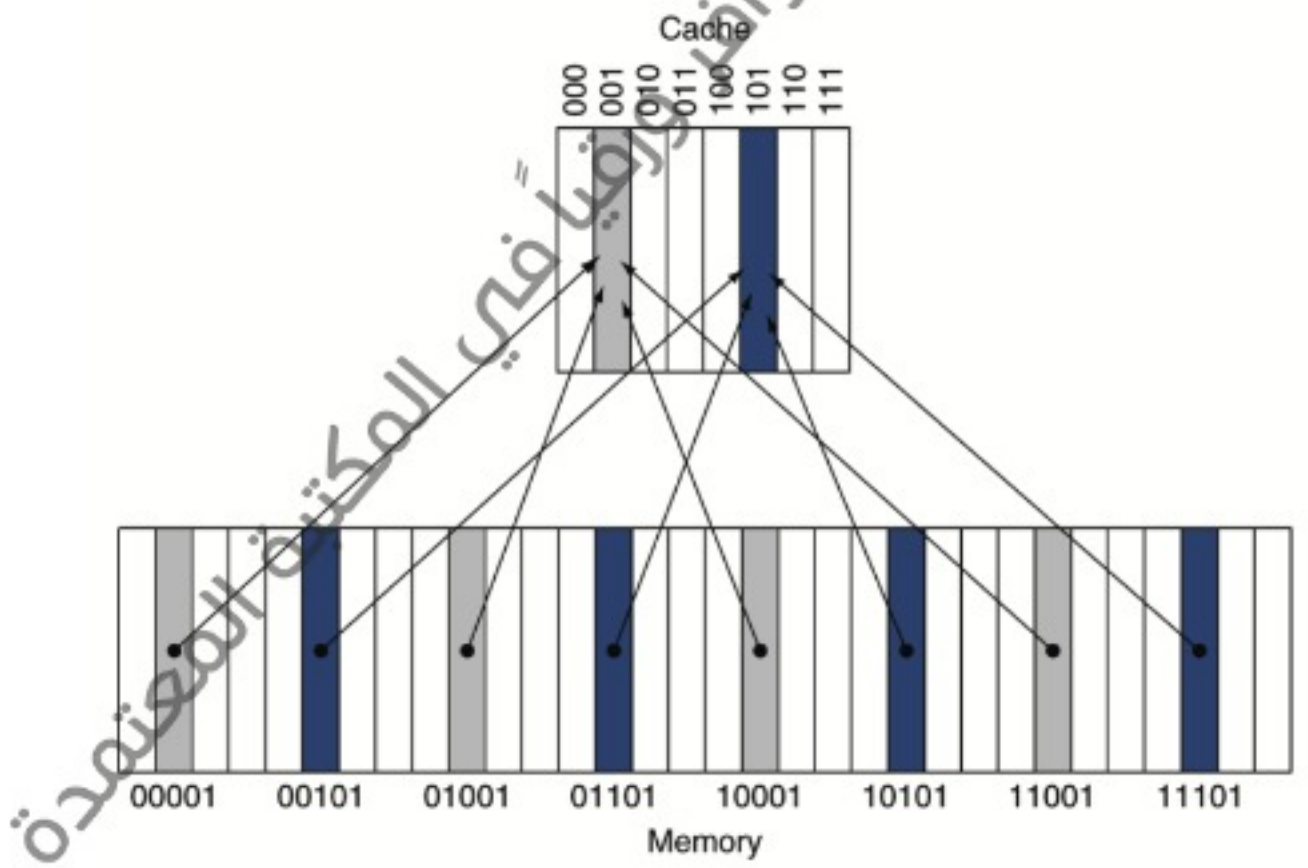
1) كل مجموعة تتضمن N من الـ Blocks.
2) ما زال العنوان يتوضع في مجموعة فريدة (محددة).
3) بيانات عنوان ما يمكنها أن تخزن أي من الـ N Blocks في المجموعة.

3. Fully associative cache:

1) يوجد مجموعة واحدة فقط.
2) يمكن للبيانات أن تذهب لأي مكان من الـ Blocks الموجودة.

Direct Mapped cache

يحدد موقعها في ذاكرة الـ cache من خلال العنوان.



يحدد موقع Block في الـ cache حسب القانون:

$$\text{قسمة باقي}\left(\frac{\text{عنوان الـ } Block \text{ في الذاكرة}}{\text{عدد } Blocks \text{ الـ } cache}\right)$$

في المثال السابق نلاحظ أن عدد الـ Blocks في الـ cache = 8، فمثلاً نريد البحث عن العناوين التالية:

:00001

و هو يمثل العدد (1)، إن باقي قسمته على 8 هو (1) و تمثيل الـ (1) في الثنائي بثلاث بتات هو 001 إذاً هي في البلوك (001).

لماذا 3 بتات؟ لأنه لدينا 8 بلوكات و نعلم أن $8 = 2^3 \Longleftarrow$ يتم تمثيل هذه المجموعة بثلاثة بتات.

:01001

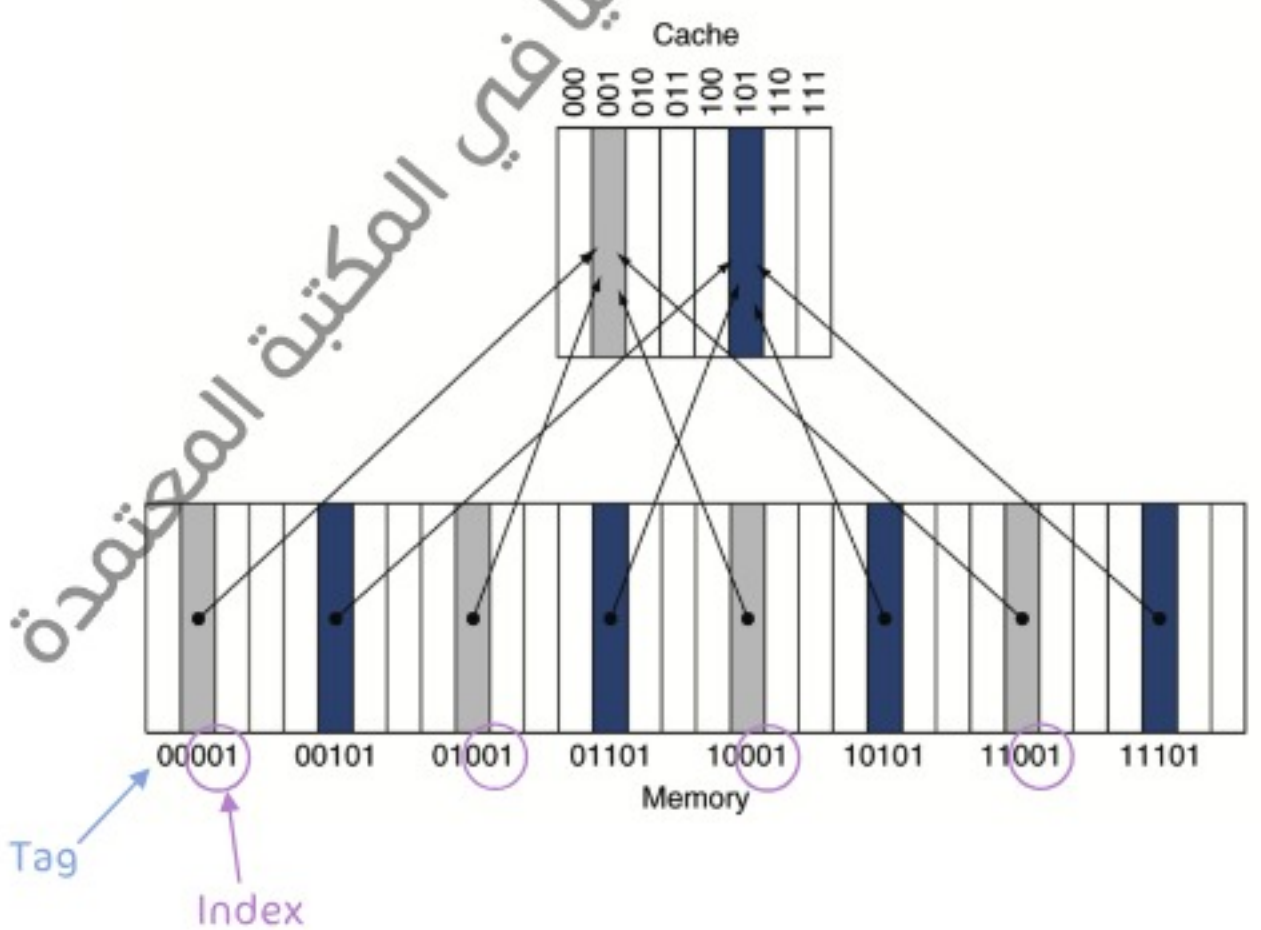
و هو يمثل العدد 9، باقي قسمته على 8 هو (1) أيضاً، و هو يوجد في العنوان (001) في الـ cache و كل عنوان باقي قسمته على 8 = 1 يوجد في نفس الـ Block (001).

:11101

و هو يمثل العدد 29، باقي قسمته على 8 هو (5) تمثيله (101) و هو الـ Block المطلوب في الـ cache.

Tags and valid bits

كيف يتم تحديد الموقع في ذاكرة الـ cache؟



فمثلاً كما في المخطط السابق، نرى أن ذاكرة الـ cache يتم تمثيل كل Block فيها بثلاثة بتات، فليكن مثلاً عنوان الذاكرة الذي نريد البحث عنه هو:

10101 10001

البتات الثلاثة لدينا تحدد عنوان الـ Block في الـ cache و تسمى: Index.
(نفس الموقع بالتأكيد بالنسبة لطريقة باقي القسمة)

البتات الباقية العليا لا علاقة لها بعنوان في الـ cache، و إنما للتمييز بين العناوين و تسمى: Tags
و ذلك لأنه يمكن أن يكون هناك أكثر من عنوان في الذاكرة له نفس العنوان في الـ cache.

- Valid bits: تحدد فيما إذا كانت المعلومة (العنوان) متاحة (موجودة) في الـ cache و تكون 1 ⟸ موجودة، 0 ⟸ غير موجودة و إن الحالة الابتدائية له هي: 0

Cache Examples:

Index	V.bit	Tag	Data
000	0		
001	0		
010	0		
011	0		
100	0		
101	0		
110	0		
111	0		

الشكل الابتدائي لجدول ذاكرة مؤقتة فيها 8 Blocks
يخزن كلمة في كل block و direct mapped

Word address	Binary address	Hit / Miss	Cache Block
$(22)_{10}$	$(10\ 110)_2$	Miss	110

في الحالة الابتدائية (أول طلب وصول للعنوان) يكون Valid bit = 0 و تكون حالة Miss

Word address	Binary address	Hit / Miss	Cache Block
26	11010	Miss	010

Index	V	Tag	Data
000	0		
001	0		
010	1	11	Mem[11010]
011	0		
100	0		
101	0		
110	1	10	Mem[10110]
111	0		

Index	V	Tag	Data
000	0		
001	0		
010	0		miss
011	0		
100	0		
101	0		
110	0		miss
111	0		

إذا أعيد طلب هذين العنوانين:

Word address	Binary address	Hit/Miss	Cache Block
22	10110	Hit	110
26	11010	Hit	010

فهما أصبحا موجودين في الـ cache.

بعد العناوين الماضية، نريد التالي:

Word address	Binary address	Hit/Miss	Cache Block
16	10000	Miss	000
3	00011	Miss	011
16	10000	Hit	000

Index	V	Tag	Data
000	Y	10	Mem[10000]
001	N		
010	N	11	Mem[11010]
011	Y	00	Mem[00011]
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Index	V	Tag	Data
000	N		miss
001	N		
010	Y	11	Mem[11010]
011	N		miss
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

نلاحظ أن عنوان الكلمة المطلوب الأول هو 16 و لم تتم قراءته من قبل فهو في حالة miss، و العنوان 3 كذلك الأمر.

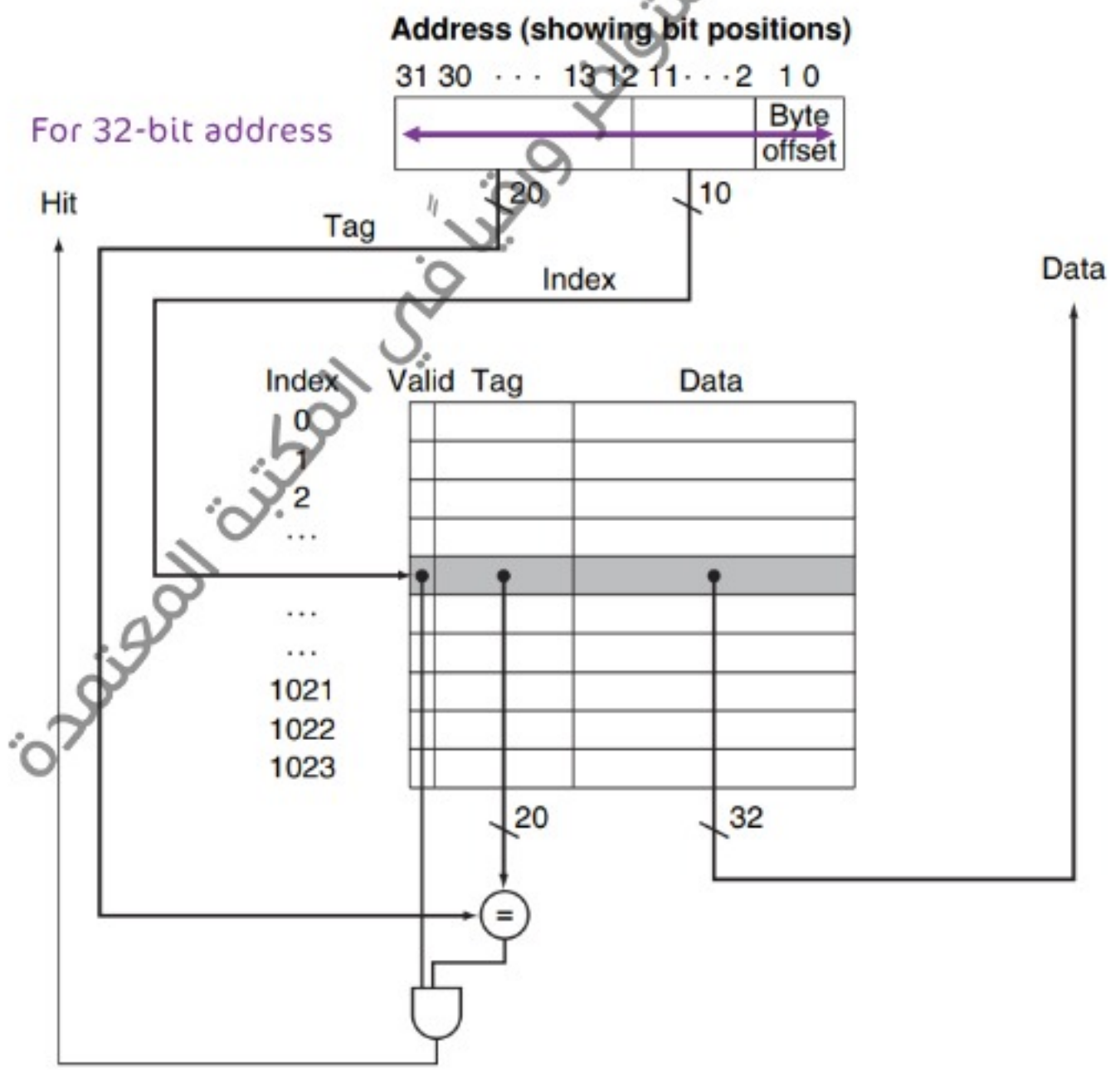
لكن العنوان الثالث هو 16 مرة ثانية، و قد تمت قراءته لذلك Hit.

Word address	Binary address	Hit / Miss	Cache Block
18	10010	Miss	010

Index	V	Tag	Data
000	Y	10	Mem[10000]
001	N		
010	Y	11	Mem[11010]
011	Y	00	Mem[00011]
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Index	V	Tag	Data
000	Y	10	Mem[10000]
001	N		
010	Y	10	Mem[10010]
011	Y	00	Mem[00011]
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

Address Subdivision



- لدينا عنوان مؤلف من 32 bits و ذاكرة direct mapped cache مؤلفة من 1024 blocks كل block فيه 1 word و كل 1 word = 4 byte فإن عدد بتات الـ offset = 2 bits، لأن 4 تمثل بـ 2 bit، عدد بتات الـ index: $\log_2(1024) = 10\ bits$
- و باقي البتات العليا لحقل الـ Tag و الـ valid bit = 1
- نقارن الـ tags باستخدام دارة طارح فإذا كان صفر كانت التاغات متساوية.

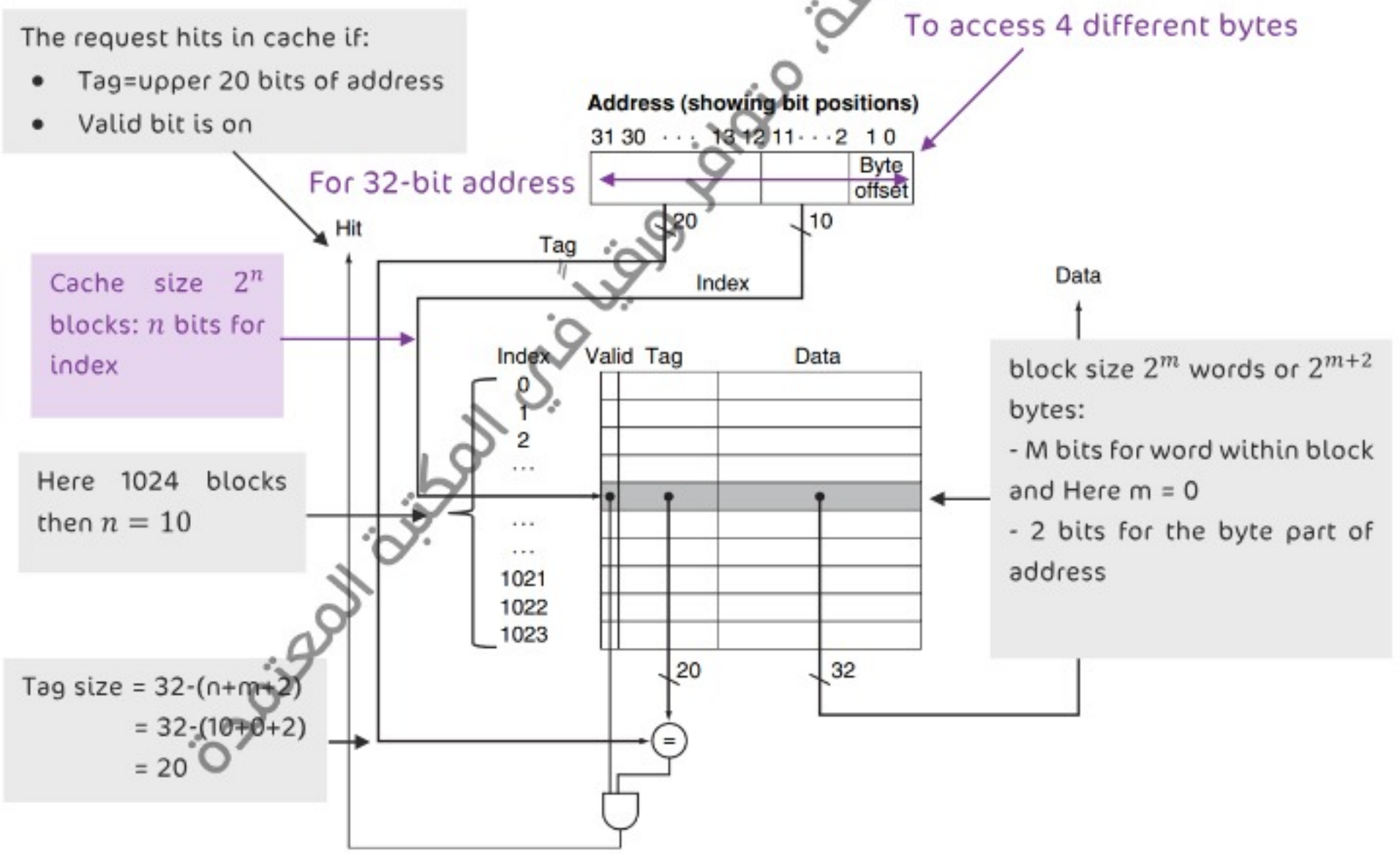
$$Total\ number\ of\ bits\ in\ cache = 2^n \times (block\ size + tag\ size + valid\ field\ size)$$

في المثال السابق:

عدد البتات الكلي في الـ cache:

Kibibits = 1024 bits

$$1024 \times (32 + 20 + 1) = 53\ kibibits$$



Total number of bits in cache:
$= 2^n \times (block\ size + tag\ size + valid\ field\ size)$
$= 1024 \times (32 + 20 + 1) = 53\ kibibits$

Example:

ما هو عدد البتات الكلي الذي تحتاجه من أجل direct mapped cache **ذات** 16 KiB **و** 4 word blocks **(كل** block **فيه** 4 words**) و بفرض العنوان ذو** 32 bits.

$$16\ KiB = 4K \times word = 4096\ words = 2^{12} words$$

$$Block\ size = 4\ words = 4 \times 4bytes = 4 \times 32\ bits = 128\ bits$$

$$\Longrightarrow m = 2$$

- لنحسب عدد الأسطر (blocks):

$$\frac{2^{12}}{2^2} = 2^{10}\ blocks$$
$$\Longrightarrow n = 10$$

- حجم الـ Tag:

$$tag\ size = 32 - (n + m + 2) = 18\ bits$$

- عدد البتات الكلية:

$$total\ bits = 2^{10} \times (128 + 18 + 1) = 2^{10} \times 147 = 147\ kibibits$$

The end...