المحاضرة الثالثة–الذاكرة المخبئية 1

بنيان الحاسب – BCA501

SLASH TEAM

برنامج الهندسة المعلوماتية – الجـــــامعة الافتراضية السورية

تقديم	مجد حاج خليل
وصف	الذاكرة المخبئية
عدد الصفحات	11

رسم خرائط الذاكرة المباشرة

مقدمة

التوصيف:

هذه المحاضرة والمحاضرتان التي تليها تركز على الذاكرة المخبئية (cache) و كيفية نقل البيانات بين الذاكرتين الرئيسية والمخبئية مع طريقة تنفيذها من قبل الهاردوير (الدارات) وتتضمن أمثلة عملية محلولة امتحانية.

طريقة الدراسة:

يتطلب دراسة هذه المحاضرة الفهم العميق فهي أساس للمحاضرتين التاليتين والتركيز على المعلومات الموجودة داخل الصور حيث أن هذه المحاضرة يبدأ فيها القسم العملي وننصح بحفظ القوانين جيدا لتسهيل الأمور في المحاضرات اللاحقة.

كيفية نقل البيانات بين الذاكرتين الرئيسية والمخبئية

عند تبادل البيانات بين الذاكرة الرئيسية والمخبئية: تكون الذاكرة الرئيسية مقسمة إلى **بلوكات** (Blocks) متساوية الحجم اما الذاكرة المخبئية تكون مكونة من **أسطر** (lines).

حجم كل سطر يساوي حجم البلوك.

الكلمة (word): هي أصغر وحدة قابلة للعنونة ضمن الذاكرة.

عندما نقول **"BYTE ADDRESSABLE MEMORY"** فهذا يعني أن حجم كل كلمة من الذاكرة هو بايت واحد.

مثال عملي عن الذاكرة الرئيسية:

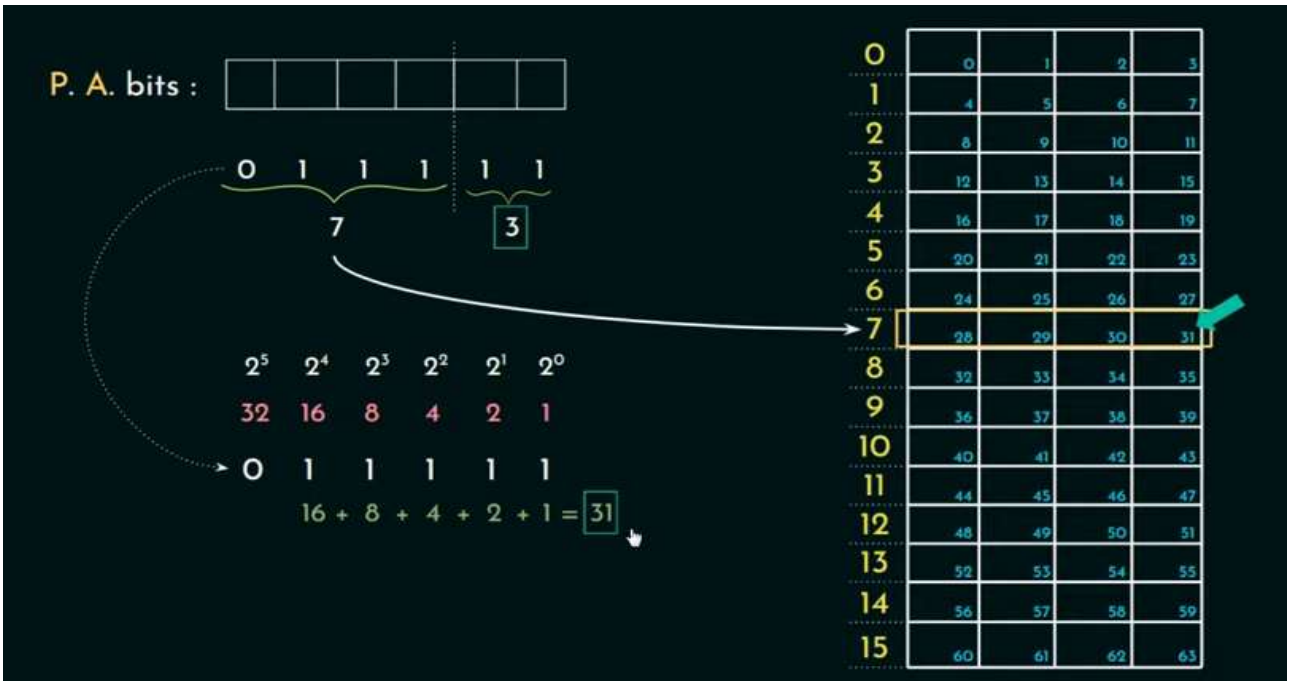
لنفرض لدينا ذاكرة رئيسية مؤلفة من 64 كلمة وحجم كل بلوك هو 4 كلمات:

يكون عدد البلوكات ضمن الذاكرة الرئيسية هو 64/4=16 بلوك.

لعنونة تلك الكلمات نحتاج إلى $log_2(64) = 6$ بتات.

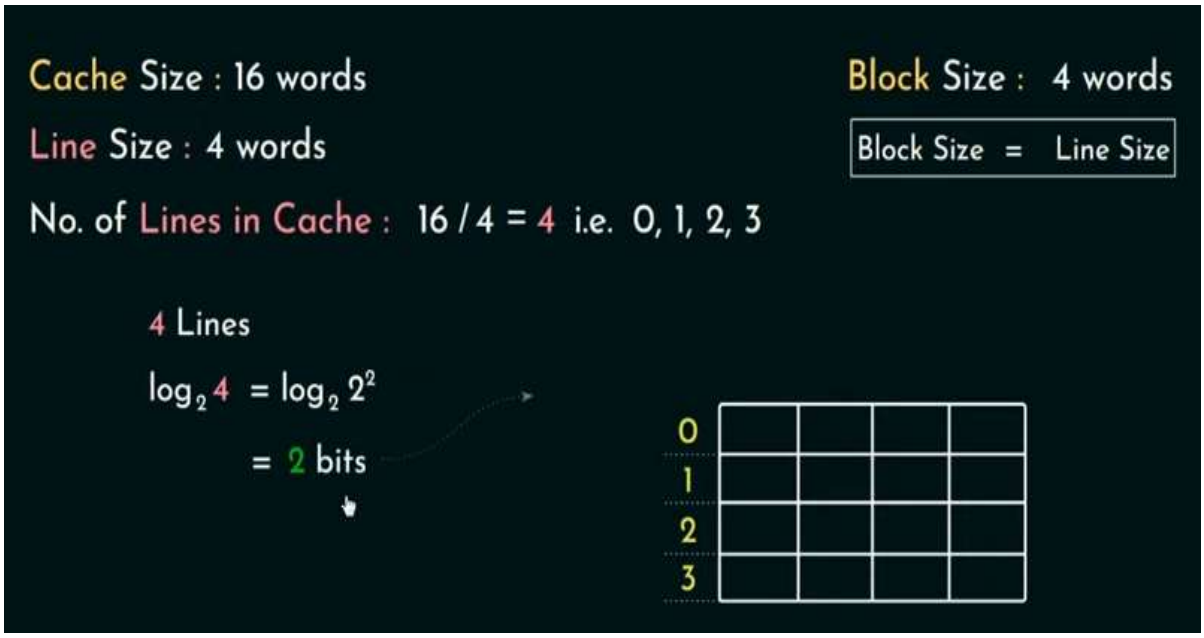
تسمى هذه البتات الستة (**P.A.BITS**) أي physical address bits.

مثال -معرفة مكان وجود الرقم 31 :



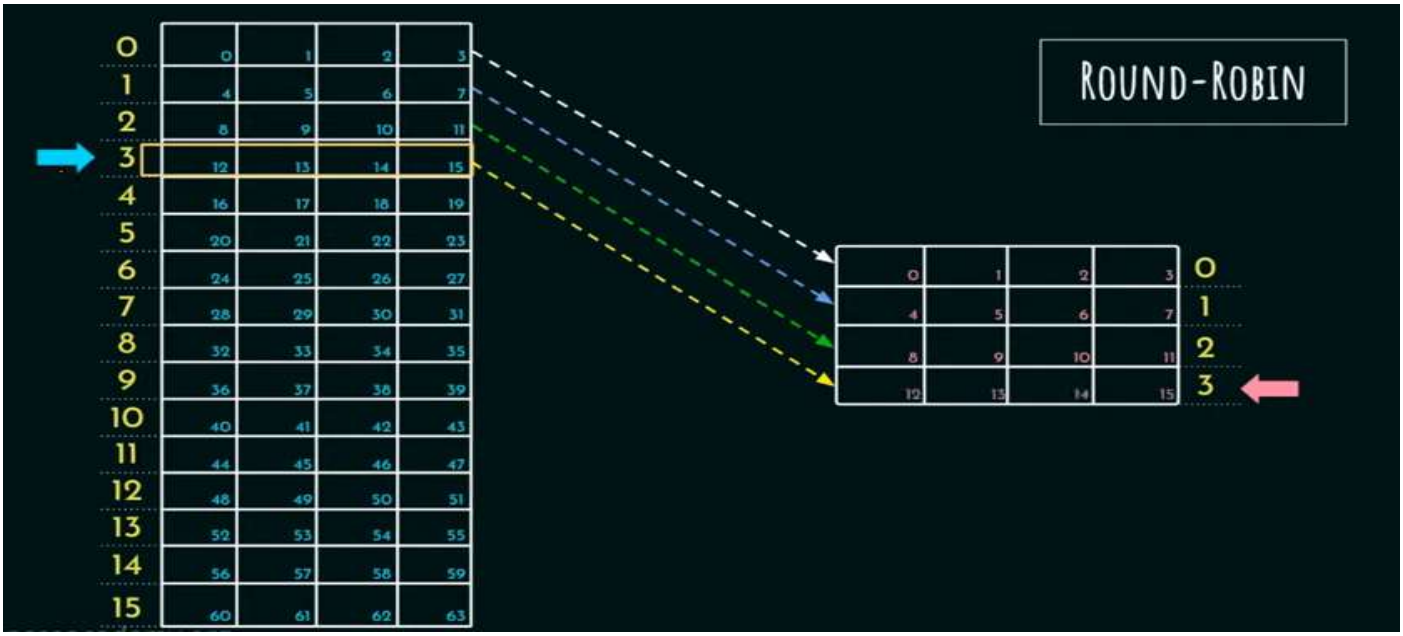
نحول الرقم 31 الى النظام الثنائي وأول أربع خانات تدل على السطر وآخر خانتين تدل على مكان وجود الرقم ضمن السطر.

مثال عملي عن الذاكرة المخبئية:



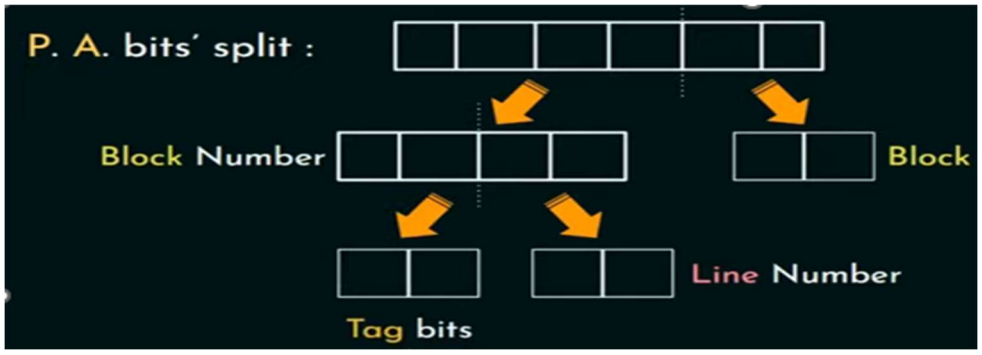
لمعرفة عدد بتات كل مربع نحسب لوغاريتم عدد الأسطر.

التقابل المباشر



في التقابل المباشر كل سطر في الذاكرة الرئيسية يقابل نفس السطر في الذاكرة المخبئية .. ثم السطر الرابع سيقابل من جديد السطر رقم 0 وهكذا .. وكل 4 أسطر (في هذا المثال)من الذاكرة الرئيسية نسميهم Tag.

نقسم ال 6 بتات الى 2 بت للبلوك و2 بت للتاغ و2 بت للسطر كما هو موضح في الصورة.



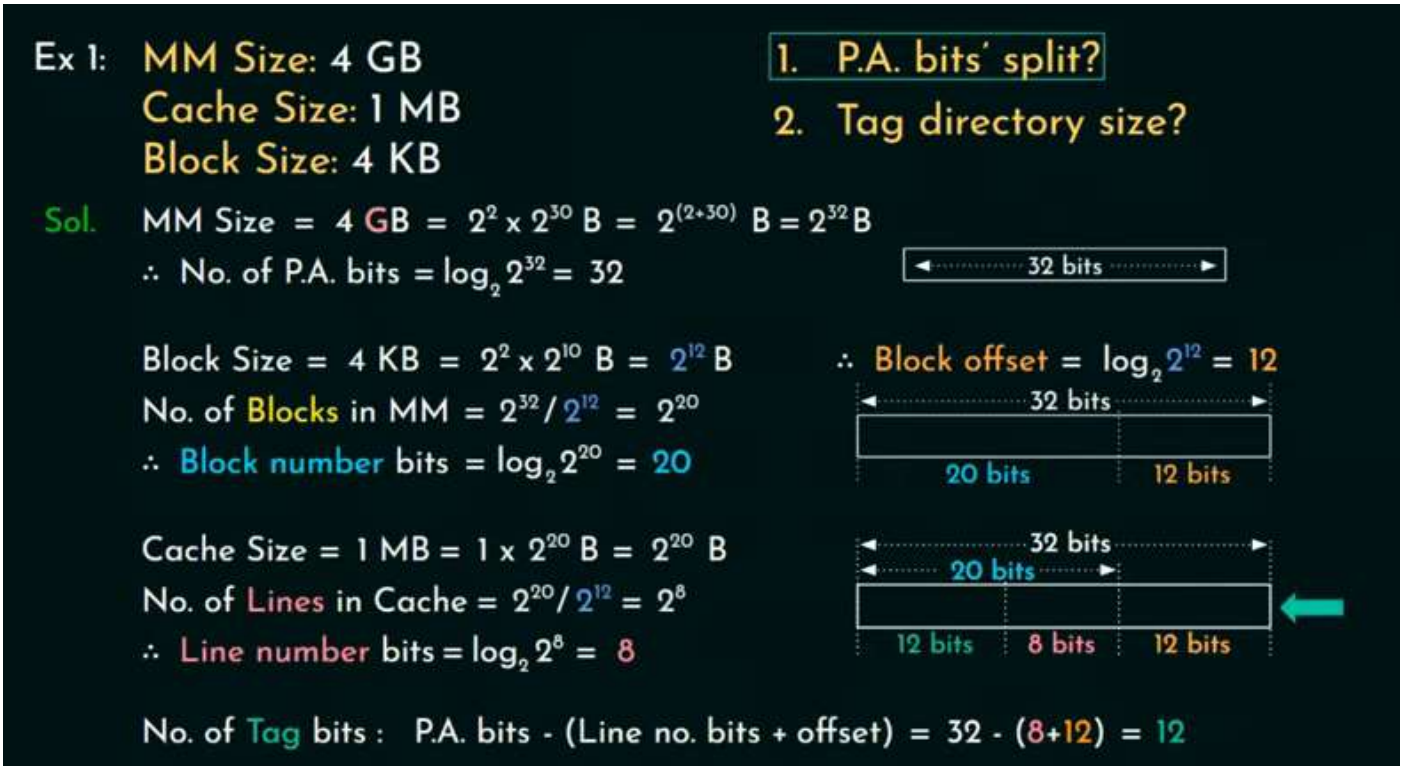
أمثلة عملية امتحانية هامة:

تحويلات ضرورية لنستخدمها في الأمثلة

1 Byte	= 8 bits	
1 KB	= 1024 B(ytes)	= 2^{10} B
1 MB	= 1024 KB	= 2^{20} B
1 GB	= 1024 MB	= 2^{30} B

المثال الأول :

الصورة تحتوي حل الطلب الأول:



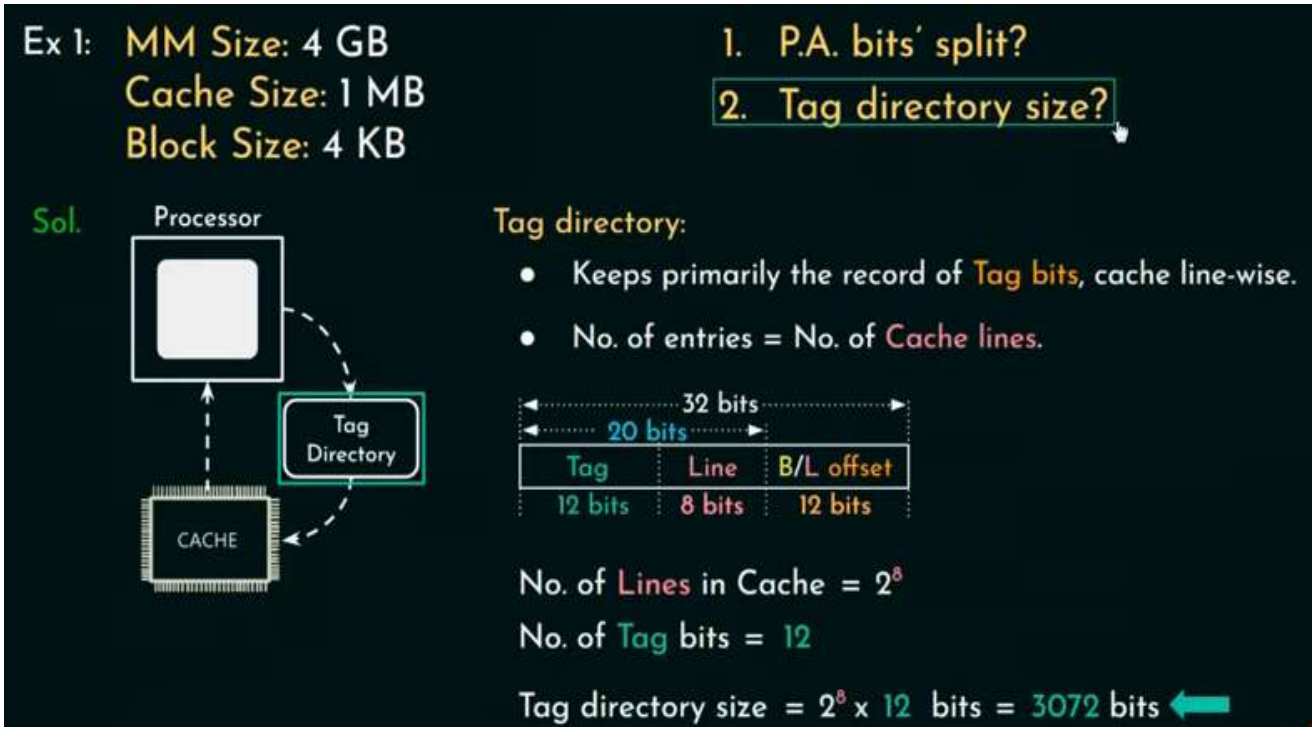
خطوات ثابتة دائما ويجب حفظها : بداية لمعرفة الحقل الفيزيائي كاملا من كم بت يتكون ونعرفه من حجم الذاكرة الرئيسية التي حجما 4GB ونحولها الى B.

لمعرفة حجم الحقل Block offset عن طريق حجم البلوك بتحويل KB4 الى B.

ولمعرفة حجم الحقل Line عن طريق عدد الأسطر ولكن في المسألة لم يعط كم سطر ولكن أعطانا حجم الذاكرة الخابية وحجم البلوك اذا نقوم بتقسيمهم وينتج لدينا $\log 2^8 \longrightarrow 2^8$

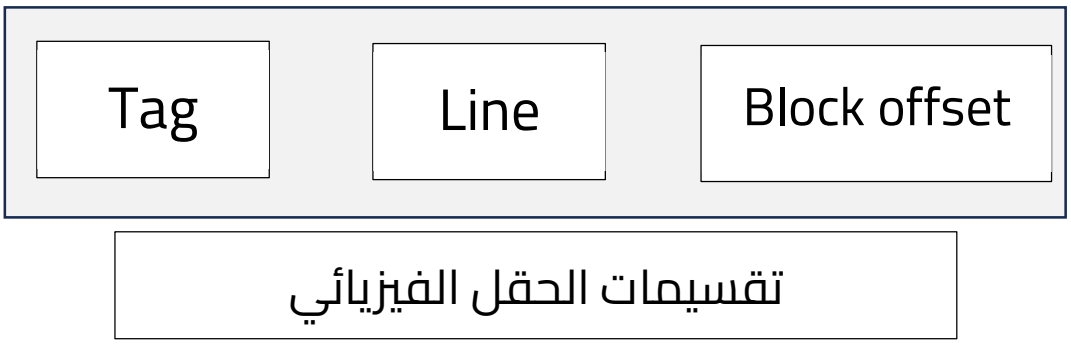
يبقى معرفة الحقل Tag نقوم بطرح حجم العنوان كاملا من الحقلين الثاني والثالث.

المطلوب معرفة حجم Tag directory:



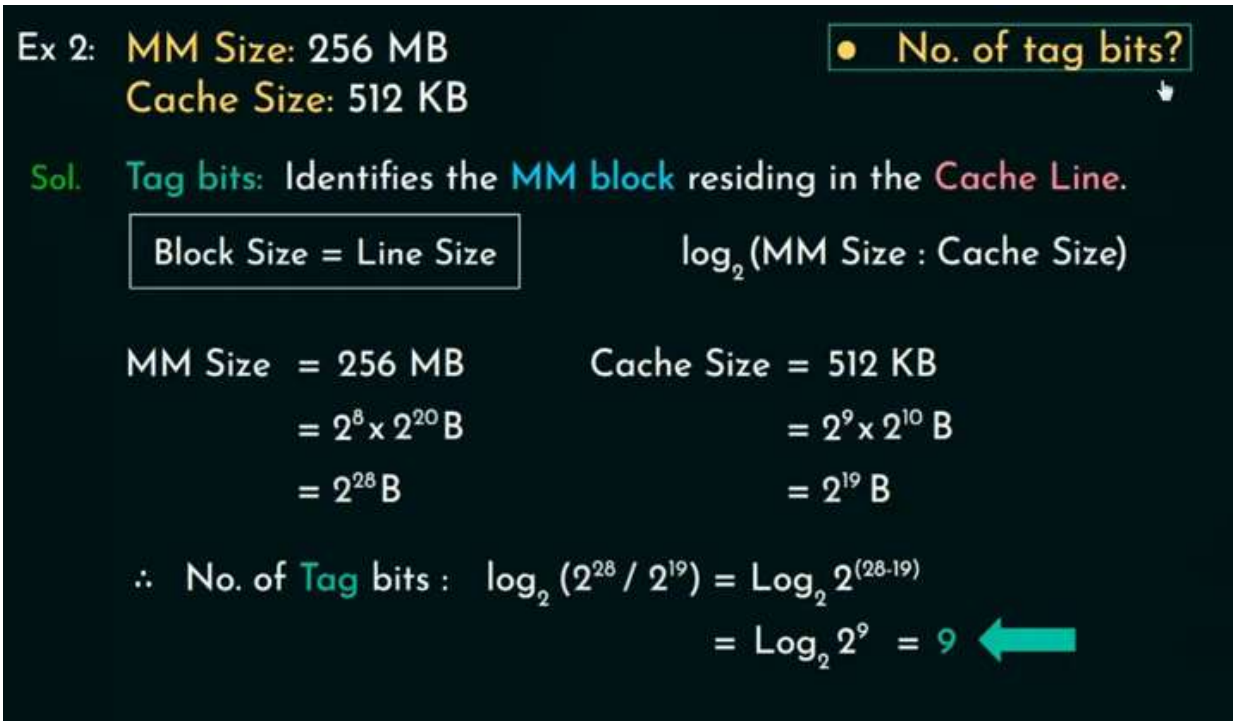
ملاحظة: المعالج عندما يريد البحث عن كلمة في الذاكرة الخابية لا يذهب اليها إنما يذهب الى Tag directory حيث يتجمع فيها كل بتات الحقل Tag ولحل هذا الطلب وبما أن عدد أسطر ال Tag مساوية لعدد أسطر الذاكرة الخابية إذا نضرب عدد الأسطر بعدد بتات ال Tag.

توضيح:



المثال الثاني:

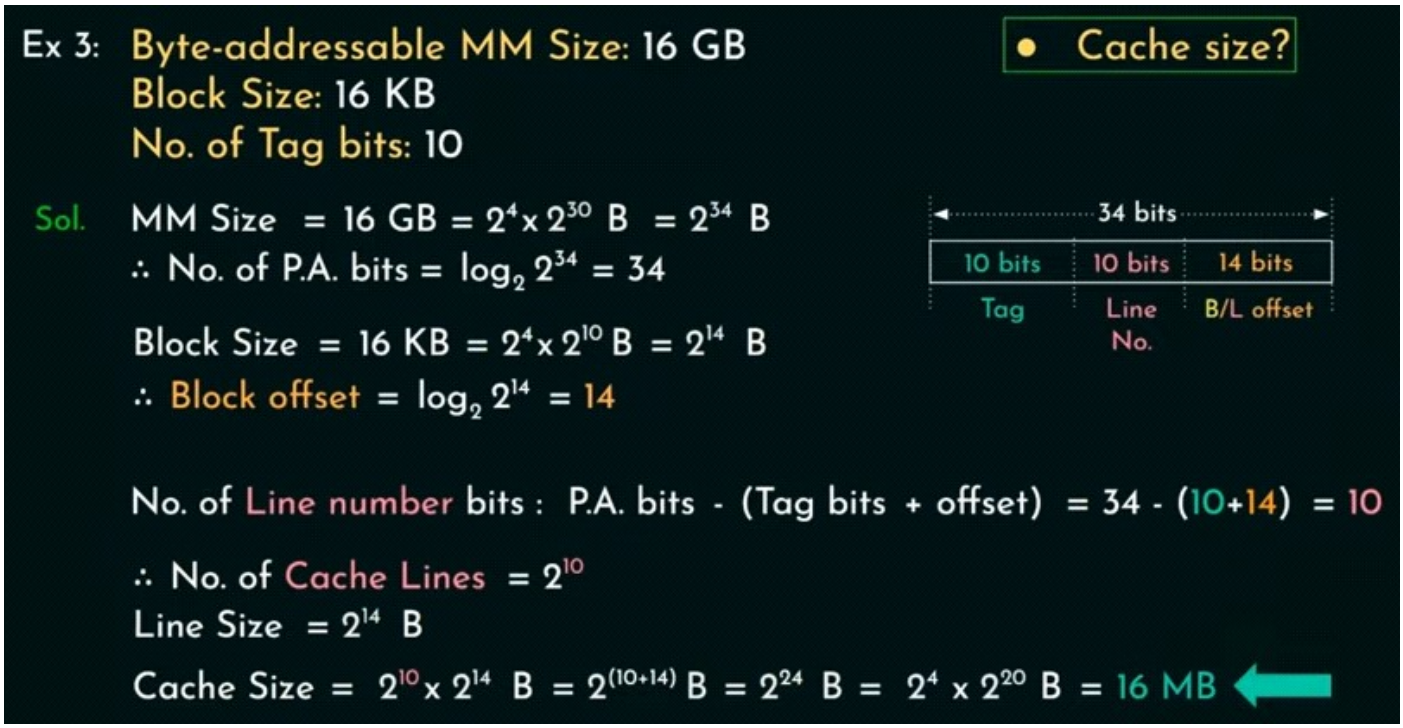
المطلوب حساب بتات حقل التاغ فقط



لحساب بتات حقل ال Tag نقسم حجم الذاكرة الرئيسية على حجم الذاكرة الخابية

المثال الثالث :

المطلوب حساب حجم الذاكرة المخبئية



ملاحظة: بداية نريد معرفة حجم العنوان الفيزيائي كاملا ونعرفه من حجم الذاكرة الرئيسية ثم لحساب حجم الذاكرة الخابية نريد معرفة حقلي ال Block offset وحقل ال Line.

لمعرفة حجم ال Block offset عن طريق ال Block size

ولمعرفة الحقل Line عن طريق طرح حجم العنوان الفيزيائي كاملا من جمع الحقلين 1 و

ولمعرفة عدد أسطر الذاكرة الخابية فقط نرفع ناتج الحقل Line على الأس 2

الآن نضرب عدد أسطر الذاكرة الخابية بحجم السطر ((block size فينتج لدينا حجم الذاكرة الخابية

التقابل المباشر عن طريق الهاردوير (الدارات)

محتويات البلوك موجودة ضمن أسطر الكاش.

رقم TAG المرافق لكل سطر من أسطر الكاش سيكون مخزنا ضمن TAG DIRECTORY

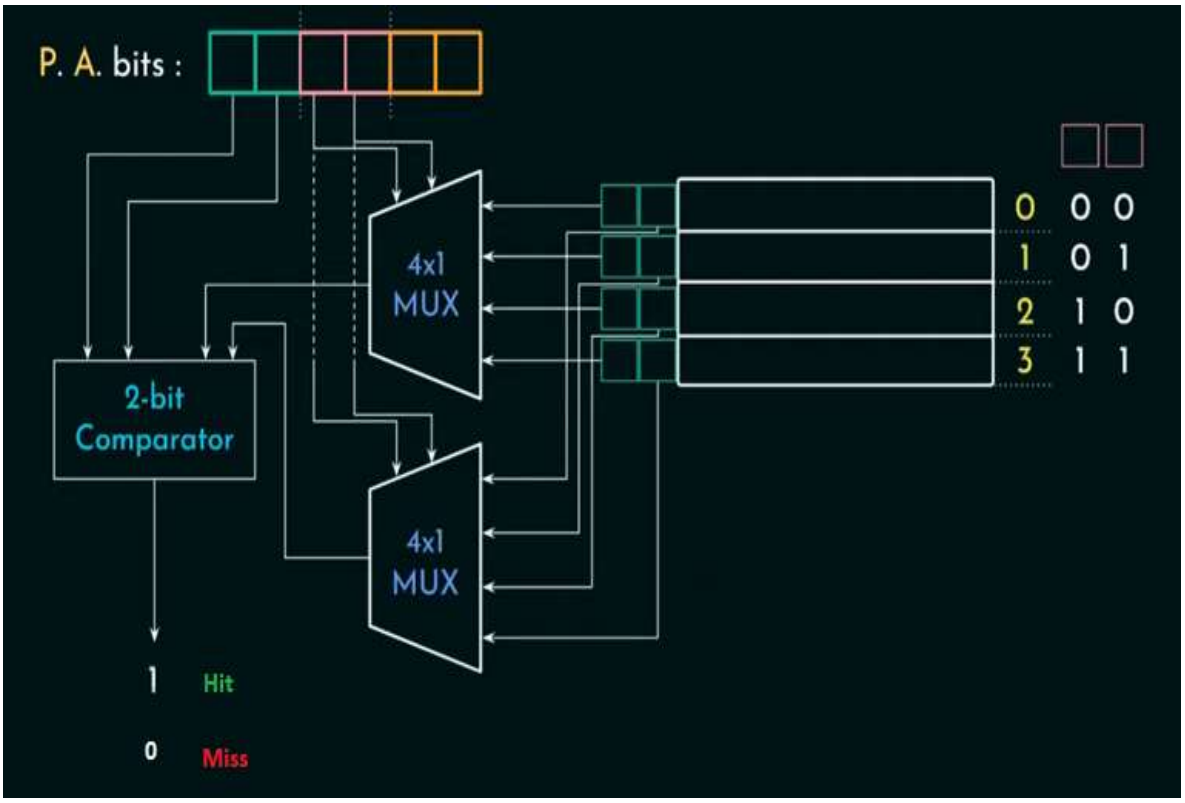
إضافة لمجموعة معلومات أخرى عن ذلك السطر مثل valid bit وغيرها.

من أجل كل بت من بتات الكاش يوجد multiplexer **عدد مداخله بعدد أسطر الكاش** وعدد مداخل select فيه يساوي عدد بتات line للعنوان الفيزيائي.

مخارج multiplexers تشكل دخل المقارن الذي هذه البتات مع بتات tag للعنوان الفزيائي المطلوب.

- إذا كان الخرج **1** فالنتيجة هي **HIT.**
- إذا كان الخرج **0** فالنتيجة هي **MISS**.

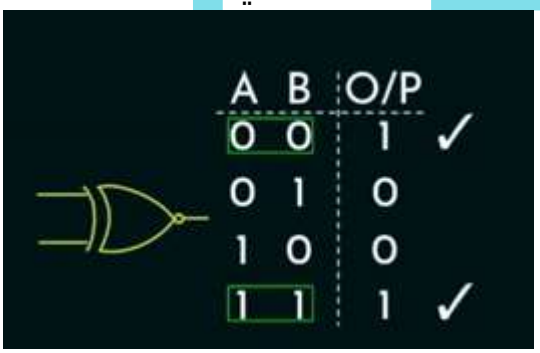
صورة لتوضيح الكلام السابق.



ما هو المقارن:

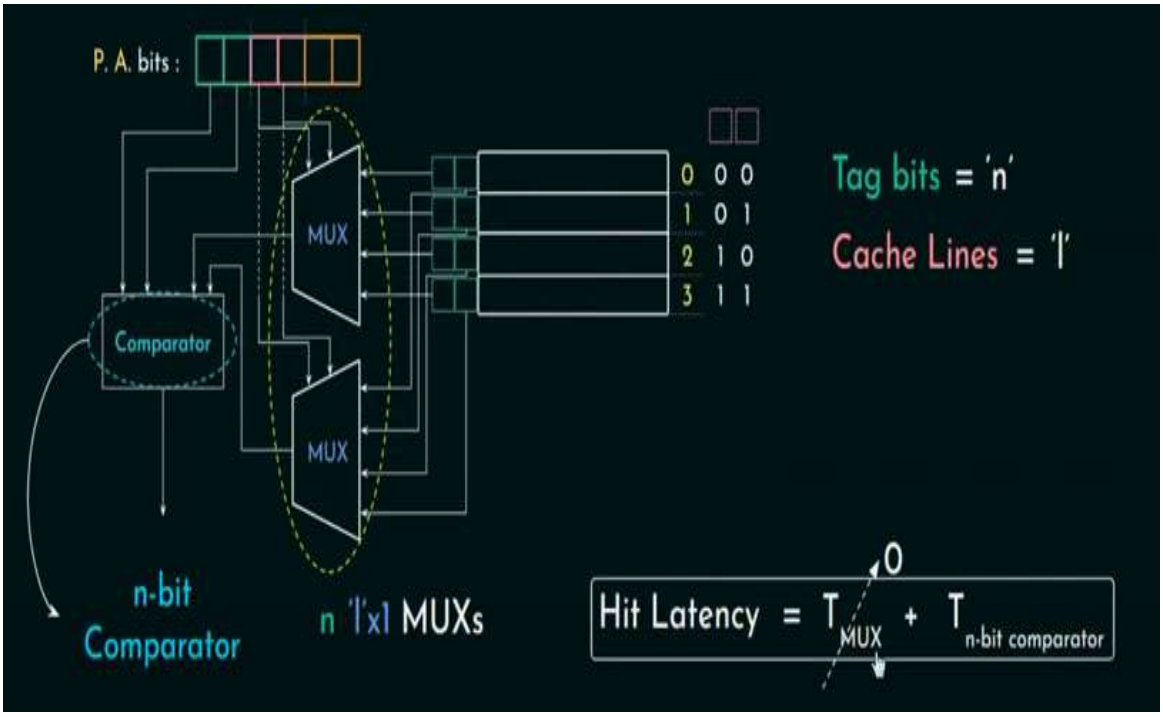
المقارن هو عبارة عن دارة EX-NOR:

- إذا كان المدخلين **متماثلين** فالنتيجة هي **1**.
- إذا كان المدخلين **مختلفين** فالنتيجة هي **0**.



من اجل حقل tag مؤلف من n بت وذاكرة كلش مؤلفة من L سطر سنحتاج إلى:

- Multiplexers: عددها n وحجمها L x 1.
- مقارن واحد بحجم n بت.



في هذه الحالة سيكون Hit latency هو:

$$\text{Hit Latency} = T_{MUX} + T_{\text{n-bit comparator}}$$

حيث:

- **TCOMPARATOR**: هو التأخير الناتج عن المقارن.
- **TMUX**: هو التأخير الناتج عن multiplexer وهو يعتبر صغيرا جدا بالمقارنة مع زمن التأخير الخاص بالمقارن لذلك يعتبر معدوما.

ملاحظة: **أخذنا Tmux واحدة لأن multiplexers تعمل على التوازي بشكل لحظي.**

لنأخذ المثال الرقمي التالي:

المطلوب التأخير اللازم عن زمن عملية البحث

Q: MM Size = 2 GB , Cache Size = 1 MB , Comparator Delay = 8n nsec
Hit latency = ?

Sol: MM Size = $2^1 \times 2^{30} = 2^{31}$ B

Cache Size = 2^{20} B

No of Tag bits = $\log_2(2^{31}/2^{20}) = \log_2(2^{31-20}) = \log_2 2^{11} = 11$

Hit Latency = 8 x 11 = 88 nsec

ملاحظة: أعطانا زمن التأخير الناتج عن مقارنة بت واحد وهو 8 nsec.

بداية يجب معرفة عدد بتات ال Tag عن طريق تقسيم حجم الذاكرة الرئيسية على حجم الذاكرة الخابية ثم نقوم بضرب زمن تأخير البت الواحد (8) بعدد البتات (11) لمعرفة زمن التأخير الكلي .

مساوئ عملية التقابل المباشر :

لنأخذ المثال التالي الذي سيوضح مساوئ هذه العملية:

إذا كانت البلوكات المطلوبة من المعالج هي:

7,8,6,10,11,14

وهذه العناوين ستتوضع عند الأسطر التالية:

3,0,2,2,3,2

نلاحظ أن السطر رقم 1غير مستخدم وسينتج لدينا cache miss من أجل السطرين 2,3. هذه المشكلة تسمى: **CONFLICT MISS**.

هذه المشكلة موضحة أكثر في المثال الرقمي التالي حيث ستتوضع جميع البلوكات ضمن السطر 2 وسيكون لدينا دائما cache miss بالرغم من كون باقي الأسطر متاحة.