المحاضرة الرابعة الذاكرة المخبئية 2

بنيان الحاسوب – BCA501

SLASH TEAM

برنامج الهندسة المعلوماتية – الجـــــامعة الافتراضية السورية

تقديم	مجد حاج خليل
وصف	الجزء الثاني من الذاكرة المخبئية.
عدد الصفحات	11

التقابل التجميعي

مقدمة

التوصيف:

هذه المحاضرة هي تكملة للمحاضرة السابقة وهي تركز على الذاكرة المخبئية (cache) حيث تكلمنا في المحاضرة السابقة عن التقابل المباشر وفي هذه المحاضرة سنتكلم عن التقابل التجميعي والتقابل التجميعي في مجموعات وتتضمن المحاضرة أمثلة عملية محلولة **امتحانية**.

طريقة الدراسة:

يتطلب دراسة هذه المحاضرة الفهم والتركيز على المعلومات الموجودة داخل الصور حيث أن هذه المحاضرة يوجد بها قسم عملي وننصح بحفظ القوانين جيدا لتسهيل الأمور في المحاضرات اللاحقة. قم بحل الأمثلة بنفسك قبل النظر للحل.

الأنواع المختلفة من CACHE MISS

تحدثنا سابقا عن conflict misses ضمن Direct mapped cache (التقابل المباشر). هذا دفع لاستخدام associative mapping (التقابل التجميعي) وبداية سنتحدث عن الأنواع المختلفة من cache miss:

COMPULSORY MISS:

وهذا ناتج عن الكتابة أول مرة ضمن الكاش. في هذه الحالة ستكون جميع البلوكات غير موجودة ضمن الكاش حالة miss يمكن التقليل من هذه الحالة من خلال زيادة عدد البلوكات ضمن السطر الواحد لذاكرة الكاش مستفيدين من مبدأ Spatial locality principle.

CONFLICT MISS

يعني أن ذاكرة الكاش كانت تضم بعض المعلومات لكن هذه المعلومات قد تم استبدالها بمعلومات أخرى خلال عملية التنفيذ. هذا النوع من الأخطاء يسمى أيضا Collision miss أو interference miss.

CAPACITY MISS

هذا النوع من الخطأ ينتج عن حجم الكاش وليس عن نوع التقابل المستخدم. تسمى البيانات اللازمة لإنجاز العمليات الحسابية الحالية باسم مجموعة العمل عندما يكون حجم هذه المجموعة أكبر من حجم الكاش فإن هذا النوع من الأخطاء سيحدث.

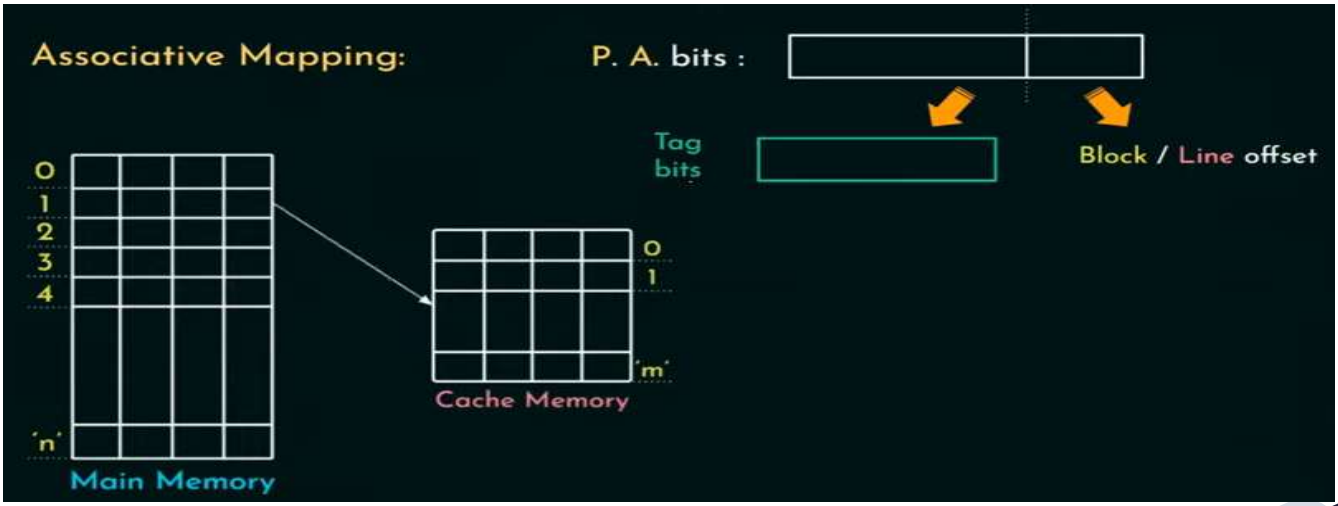
تسمى هذه الأنواع الثلاثة من الأخطاء باسم CS3. ويعتبر تحديد النوع الثالث من miss هو الأصعب.

إضافة للأنواع الثلاثة السابقة توجد أنواع أخرى من cache miss سنتحدث عنها لاحقا منها:

- coherence miss
- Coverage miss
- System related miss

الحل لمشكلة CONFLICT MISS هو ASSOCIATIVE MAPPING

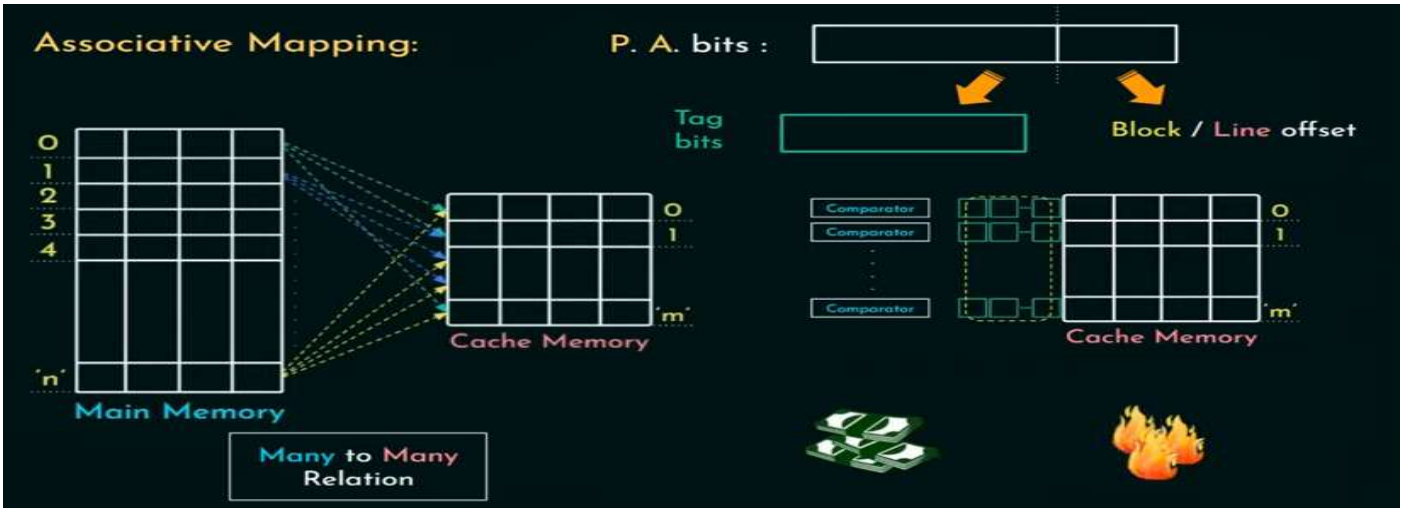
في حالة Associative mapping لا توجد تقييدات على تقنيات التقابل. أي يمكن تخصيص أي سطر من أسطر الكاش من أجل أي بلوك من بلوكات الذاكرة الرئيسية. يقسم العنوان الفيزيائي إلى حقلين block offset وblock numberg. يتضمن حقل block number ضمنيا بتات tag.



لكن هنا ستظهر إشكالية أخرى:

بما أن عملية التقابل هذه تتضمن علاقة من نوع many-to-many relationship فلا يوجد لدينا أي دليل أين سيكون موقع أي بلوك من الذاكرة الرئيسية ضمن أسطر الكاش. وبالتالي خلال عمليات الاستعادة (retrieve) يجب فحص بتات tag لجميع أسطر الكاش وهذا يزيد hit latency.

في هذه الحالة يمكن استخدام مقارنات بعدد أسطر الكاش لكن ذلك سيزيد تعقيد الهاردوير بشكل أسي كما يزيد من الحرارة.



ويصبح لدينا الان :

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

مثال -المطلوب حساب زمن التأخير:

Q: MM Size = 2 GB

Block Size = 2 KB

Comparator Delay = 15n nanoseconds.

Delay of Multi-input OR gate = 7 nanoseconds.

• Hit latency = ?

Sol. MM Size = $2^1 \times 2^{30} = 2^{31}$ B ∴ P.A. bits = 31

Block Size = 2^{11} B ∴ Block offset = 11

No of Tag bits = (31 - 11) = 20

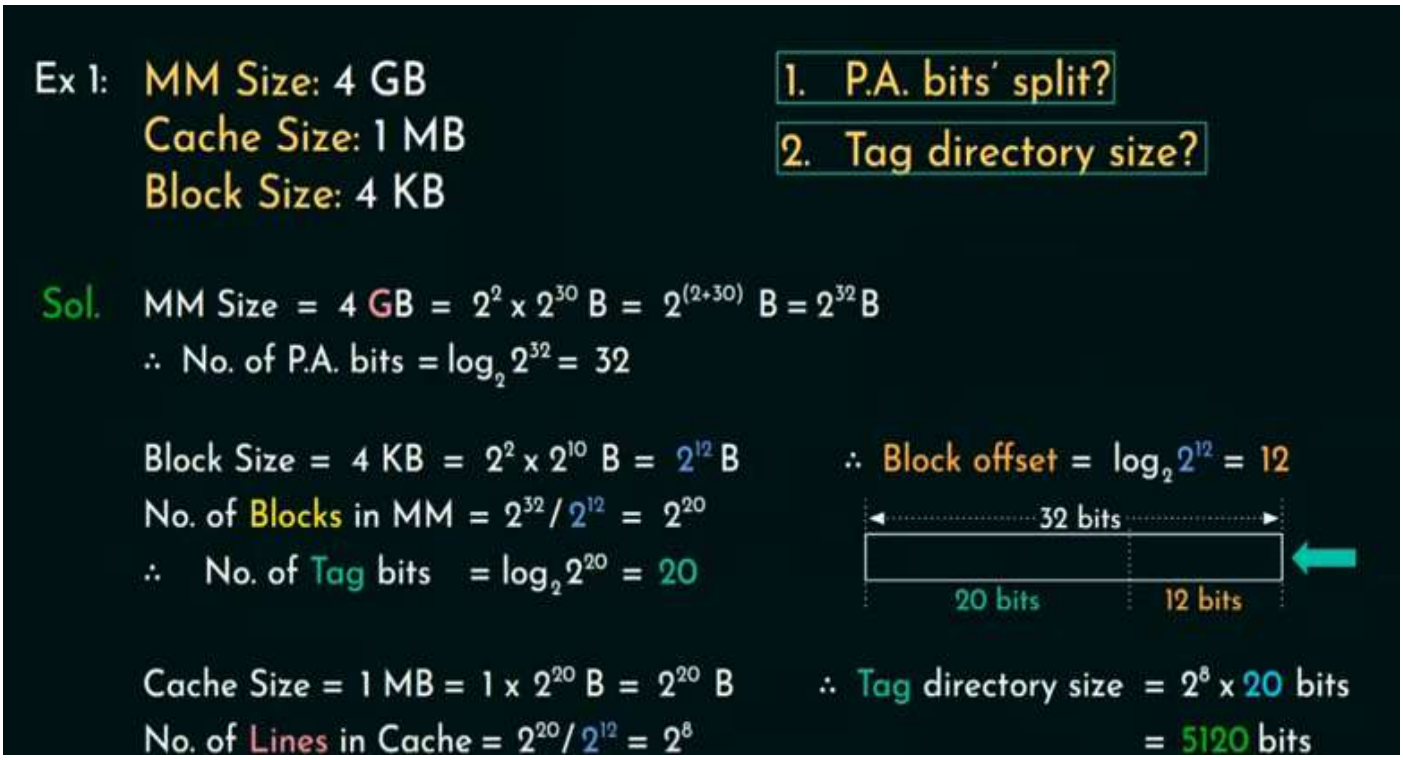
Hit Latency = (15 x 20) + 7 = 307 nsec

طريقة الحل:

حسب القانون فإن زمن التأخير الكلي هو حاصل **جمع زمن التأخير الناتج عن المقارن مع زمن التأخير الناتج عن OR** ولكن زمن التأخير الناتج عن المقارن معطى حسب بت واحد من بتات Tag إذاً بداية نحسب حجم حقل Tag عن طريق معرفة شكل العنوان وفي التقابل التجميعي هو حقلين أحدهما word والأخر Tag .. والان لمعرفة حجم العنوان كاملا نحسبه من حجم الذاكرة الرئيسية ولحساب حجم حقل ال word نحسبه من حجم حقل Block ويبقى لدينا حقل Tag هو تحصيل للعملية السابقة .. الان عرفنا عدد بتات ال Tag فنحسب زمن تأخير المقارن بالنسبة لكل عدد البتات ونجمع معه زمن التأخير الناتج عن OR فنحصل على زمن التأخير الكلي.

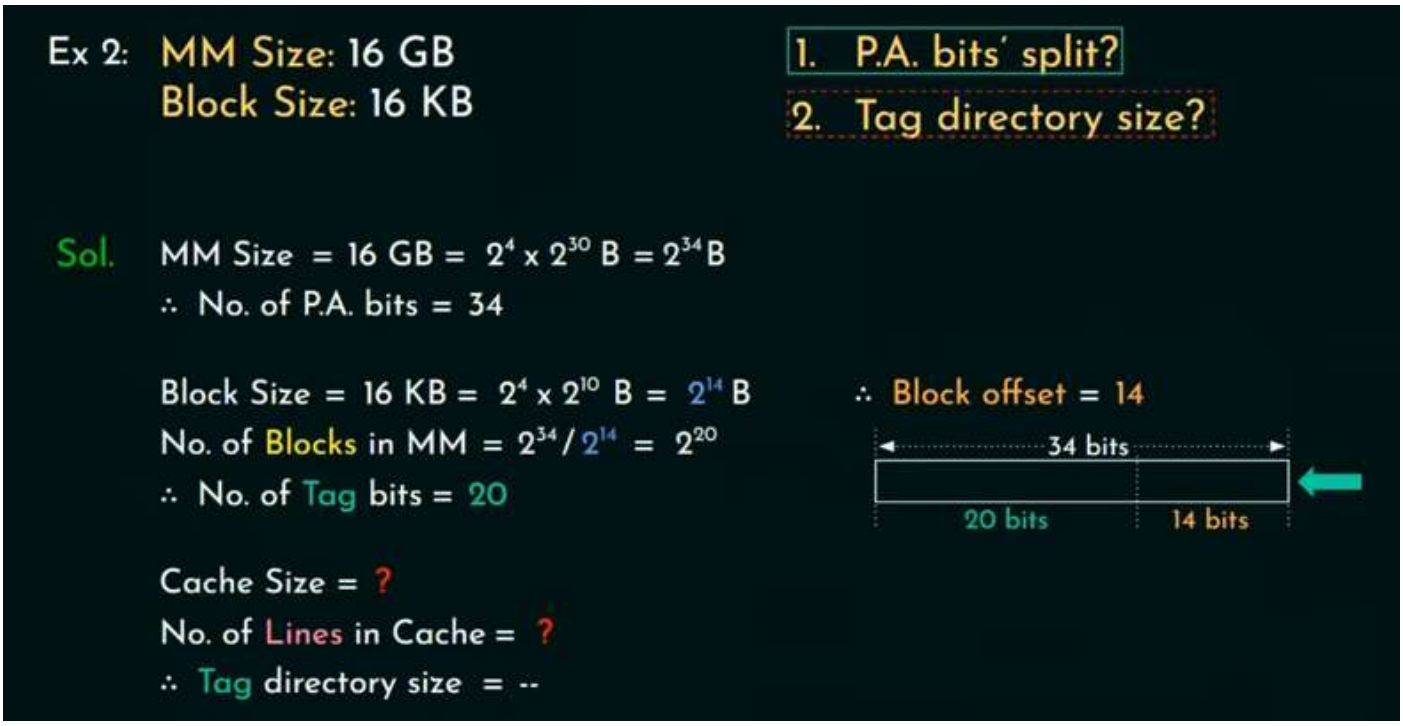
ملاحظة: التأخير المعطى عن المقارن هو 15n لكل بت في Tag bits. **فزمن التأخير** الكلي الناتج عن المقارن هو مجموع زمن التأخير في كل بت أي **Comparator delay * Tag bits**

مثال – مشابه للأمثلة السابقة بنفس طريقة الحل:



الطلب الثاني بخصوص الحجم الكلي الذي تشغله Tag bits. هو عدد أسطر ذاكرة الكاش مضروب بعدد بتات Tag.

مثال:



طريقة الحل: نحسب حجم العنوان كاملا من حجم الذاكرة الرئيسية ثم نحسب حجم حقل word من حجم block ويبقى لدينا حقل Tag هو تحصيل للعملية السابقة . الان لحل الطلب الثاني بنفس قانون التقابل المباشر تماما أي نأخذ عدد حقل بتات ال Tag ونضربه ب عدد أسطر الذاكرة الخابية ولكن في هذه المسألة لم يعطنا عدد أسطر الذاكرة الخابية ونحن عادة نستنجها من حجم الذاكرة الخابية عن طريق تقسيم حجمها على حجم Block ولكن لم يعطها في هذه المسألة - المعطيات ناقصة.

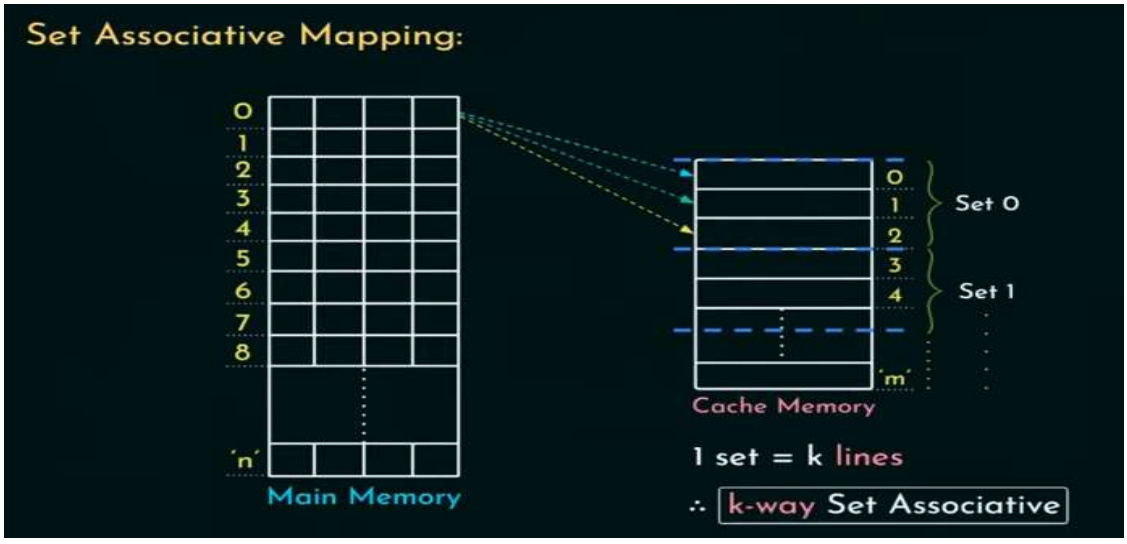
التقابل التجميعي في مجموعات

في هذه الطريقة:

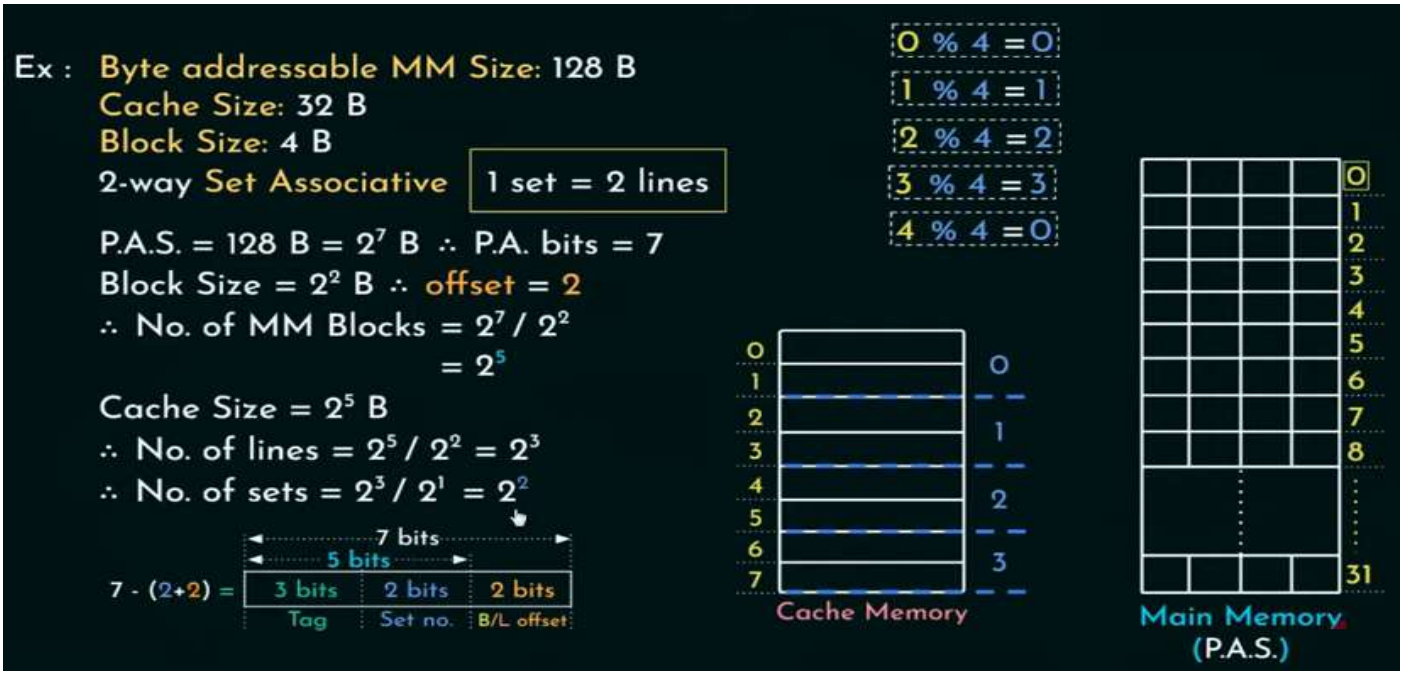
- تكون الذاكرة المخبئية مقسمة إلى مجموعات (Sets) من الأسطر.
- كل بلوك من الذاكرة الرئيسية يكون في مجموعة محددة من مجموعات الذاكرة المخبئية.

لكنه يمتلك المرونة لأن يكون في أي سطر من أسطر المجموعة الواحدة من مجموعات الذاكرة المخبئية في هذه الحالة - إذا كانت المجموعة الواحدة تتضمن K سطرا:

نطلق على تقنية التقابل هذه اسم K-way Set associative mapping: نحتاج فقط إلى K مقارن

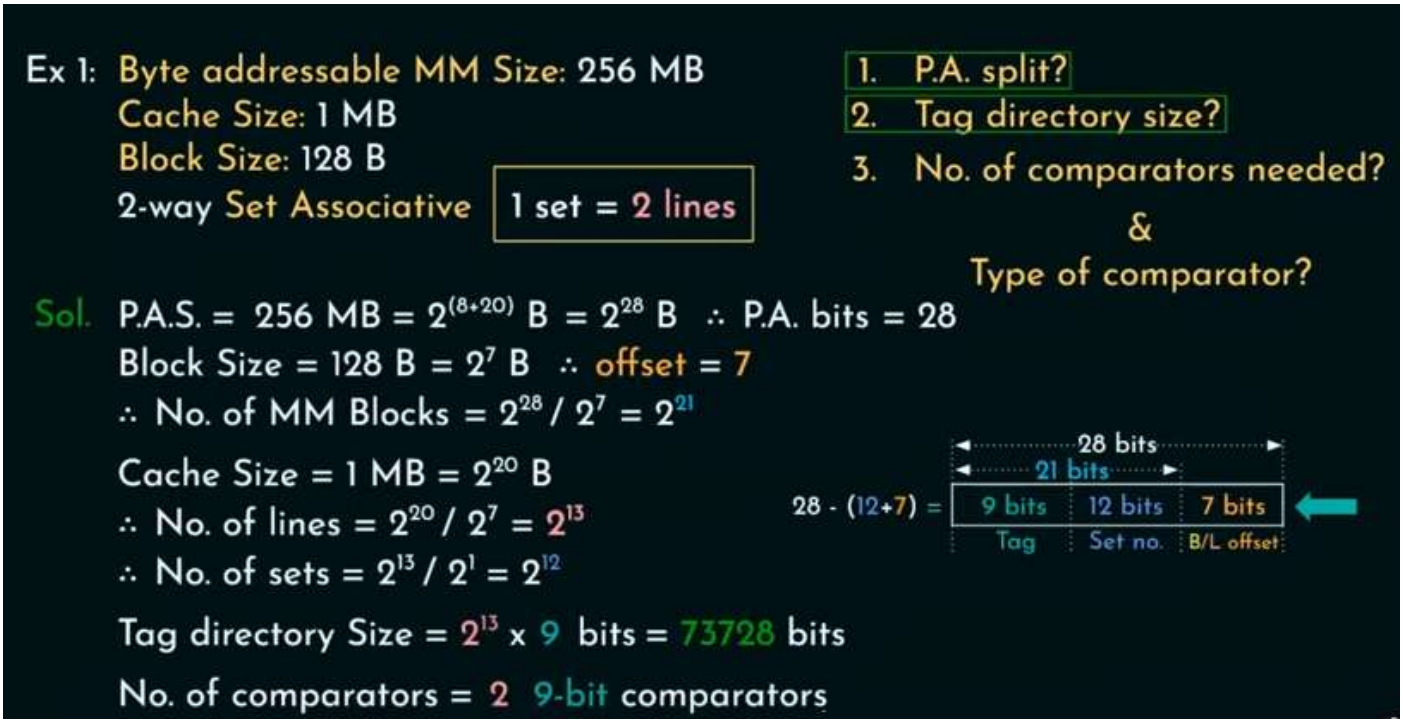


مثال:



طريقة الحل: بداية عندما يقول 2 -way إذا نحن امام مسألة تقابل مجموعات إذا نحن بحاجة لمعرفة مكان الكلمة وبأي مجموعة هي وليس بأي سطر بالإضافة الى لزوم معرفة حجم Tag. الان لمعرفة حجم العنوان كاملا من حجم الذاكرة الرئيسية ثم لمعرفة حجم الحقل Word من حجم Block ولمعرفة أي مجموعة نعرفه من عدد المجموعات في الذاكرة الخابية ولمعرفة عدد المجموعات نستنتجه من تقسيم حجم الذاكرة الخابية على حجم Block والان لدينا كل سطرين بمجموعة ولدينا 8 أسطر وكل 2 ضمن مجموعة إذا لدينا 4 مجموعات ويبقى لدينا حقل Tag هو تحصيل للعملية السابقة.

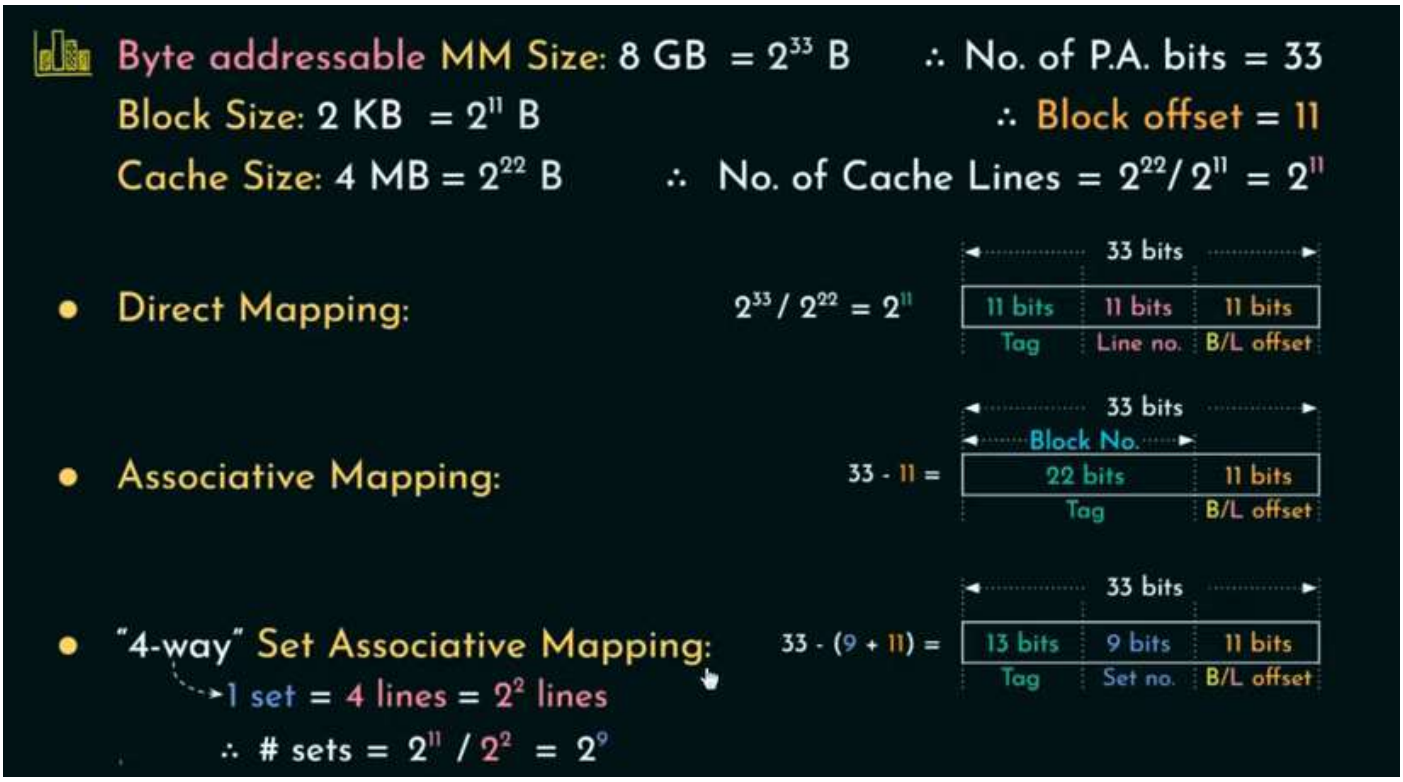
مثال:



طريقة الحل: الطلب الأول والثاني مشابه تماما للطلبات السابقة اما بالنسبة للطلب الثالث حيث طلب معرفة عدد المقارنات الموجودة ولمعرفتها نتبع القاعدة التالية:

إذا كان لدينا سطرين في المجموعة اذن نحتاج مقارنين فقط وإذا كان لدينا كل 4 أسطر في المجموعة اذن نحتاج 4 مقارنات وهكذا .. وكما أنه طلب نوع المقارن ويمكننا معرفة نوعه من حجم حقل Tag وهو 9 bit.

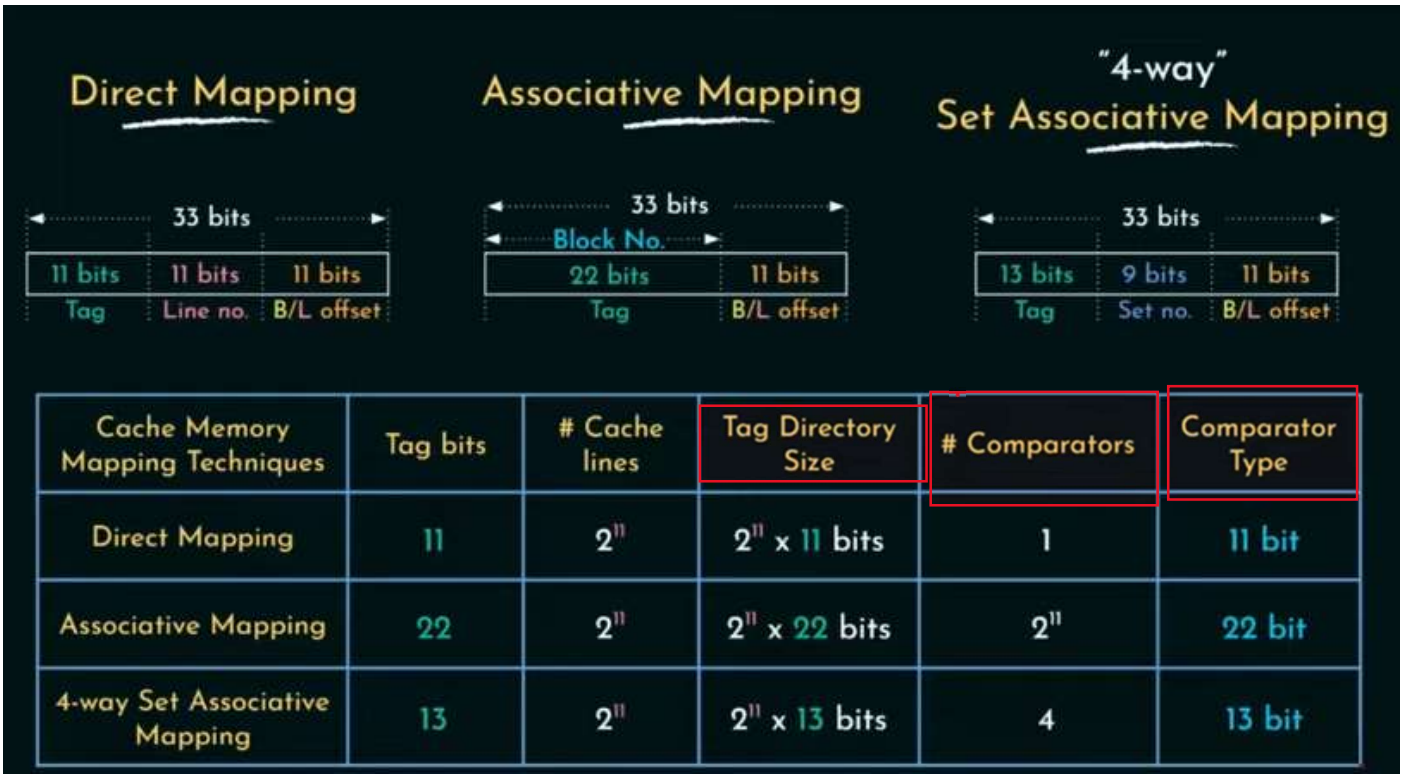
مثال شامل لأنواع التقابل الثلاثة:



طريقة الحل: حجم العنوان الفيزيائي الكامل ثابت في الطرق الثلاثة .. وقد حللنا امثلة سابقة عنهم ولكن للتذكير بداية كنا نحسب حقل word من حجم block ونحسب حقل Line من عدد الاسطر ونستنتج عدد الأسطر من تقسيم حجم الذاكرة الخابية على حجم السطر.

الان في التقابل التجميعي لدينا فقط حقلين wordg tag ولحساب حقل word عن طريق حجم Block .. الان في التقابل التجميعي في المجموعات أيضا ثلاث مجموعات هم set, tag, word ولدينا حجم حقل ال word ثابت في كل المسألة ولحساب حجم حقل ال set نعرفه من عدد المجموعات حيث لدينا كل 4 أسطر في مجموعة إذا نقسم عدد الاسطر على عدد المجموعات فنعلم حجم حقل Set ويبقى لدينا حقل Tag هو تحصيل للعملية السابقة.

تكملة للمثال السابق:

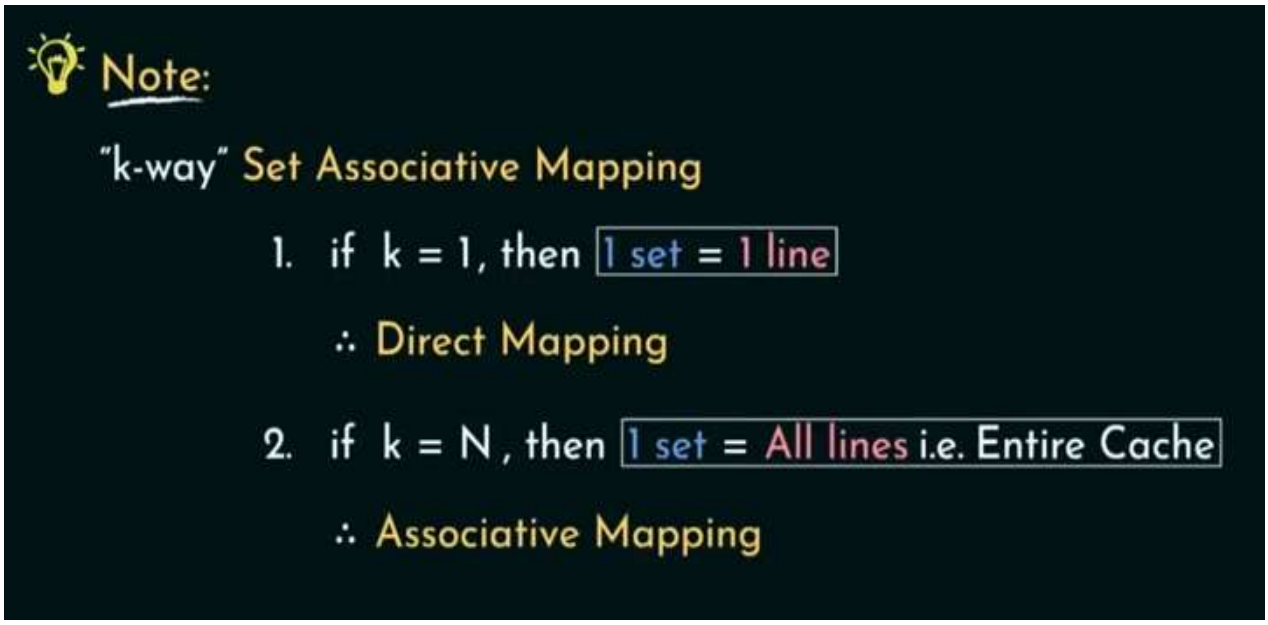


Cache Memory Mapping Techniques	Tag bits	# Cache lines	Tag Directory Size	# Comparators	Comparator Type
Direct Mapping	11	2^{11}	2^{11} x 11 bits	1	11 bit
Associative Mapping	22	2^{11}	2^{11} x 22 bits	2^{11}	22 bit
4-way Set Associative Mapping	13	2^{11}	2^{11} x 13 bits	4	13 bit

طريقة الحل: لحساب حجم المجمع الخاص ب Tag لدينا قانون وهو عدد الأسطر مضروب بعدد بتات ال tag ونأخذ المعطيات من حل المسألة السابقة.

- والان لحساب عدد المقارنات في التقابل المباشر لدينا دائما مقارن واحد.
- وفي التقابل التجميعي عدد المقارنات هو نفسه عدد الأسطر.
- بينما في التقابل التجميعي في مجموعات فإن عدد المقارنات هو حسب عدد الأسطر ضمن المجموعة الواحدة.. وكما أنه طلب نوع المقارن ويمكننا معرفة نوعه من حجم حقل Tag.

فكرة أخيرة في التقابل في مجموعات :



طريقة الحل: إذا كان لدينا عدد الأسطر الموجودة في المجموعة الواحدة سطر واحد فقط إذاً عدنا الى النوع الأول من أنواع التقابل وهو التقابل المباشر **وإذا** كان لدينا عدد الأسطر التي سنضعها في المجموعة الواحدة هي كل أسطر الذاكرة الخابية إذا نحن عدنا الى نوع التقابل التجميعي.