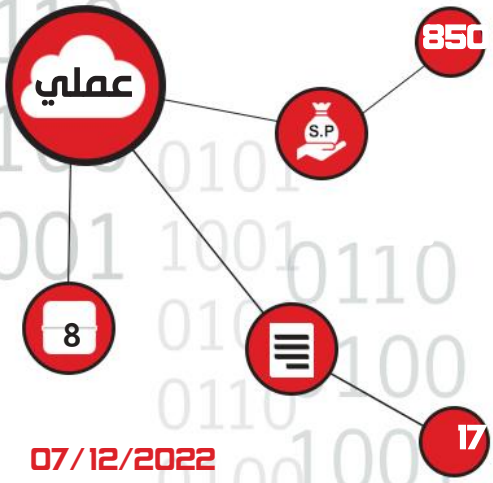


الذاكرة الخابية
Cache Memory

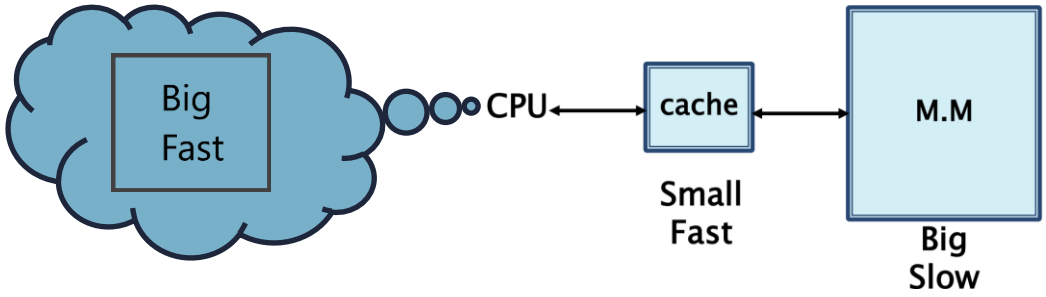
م. عبير ميّا م. مصعب خباز

07/12/2022

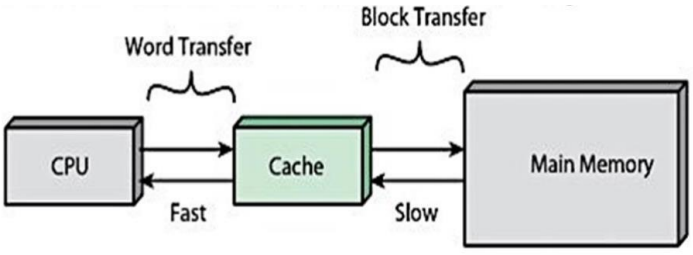
▪ الذاكرة الخابية

وهي عبارة عن ذاكرة صغيرة ولكنها سريعة جداً, وبما أن جلب التعليمة من الذاكرة هي عملية مكلفة كثيراً للوقت نقوم بإضافة هذه الذاكرة $cache$ بحيث تكون جسر الوصل بين المعالج والذاكرة.

$$small\ fast\ memory + big\ slow\ memory \quad \Leftrightarrow \quad looks\ like\ a\ big\ fast\ memory$$



- **من المهم أن نعلم أنه يتم التعامل بين المعالج والـ$cache$ بالكلمة $(word\ transfer)$ حيث يقوم بطلب كلمة واحدة في كل عملية بينما يتم التعامل بين الذاكرة والـ$cache$ عن طريق كتلة $block$ (كتلة من الكلمات) لأن عملية الجلب مكلفة فنقوم بجلب كتلة كاملة عوضاً عن التعامل بالكلمة.**



- **عندما يطلب المعالج محتوى عنوان بالذاكرة يوجد حالتين:**

1. **المحتوى موجود في الـ$cache$: وهي حالة إصابة (Hit) فيقوم بالحصول على المحتوى بسرعة (سرعة حصول على المعطيات).**
2. **المحتوى غير موجود في الـ$cache$: وهي حالة إخفاق $(Miss)$ حيث يجب الذهاب إلى الذاكرة الرئيسية وجلب $block$ وتخزينها في الخابية $(cache)$ ثم يحصل المعالج على المحتوى المطلوب.**

- تضم الخابية حقل الأمارة tag ومن خلاله يتم تحديد فيما إذا كانت الكتلة المطلوبة موجودة بالـ$cache$ أم لا وسنقوم بشرح حقول التعليمة بالتفصيل فيما بعد...

مبدأ عمل الخابية (Cache)

يعتمد مبدأ عمل الخابية على المحلية الزمانية والمكانية:

1. المحلية المكانية *Spatial Locality*: طلب المعطيات المجاورة.

مثال: تخزين خاناتها بعناوين متتالية في الذاكرة وإذا طلب المعالج أحد خانات هذه المصفوفة فتقوم الذاكرة الخابية بأخذ كامل الكتلة التي تحتوي على الخانة التي طلبها المعالج من الذاكرة *RAM* ولكن لماذا؟ 🤔

لأنه إذا طلب المعالج أحد هذه الخانات فإنه مستقبلاً سيطلب أحد الخانات التالية وبذلك مستقبلاً نوفر على الخابية أخذ خانات أخرى من المصفوفة من الذاكرة RAM في كل مرة يطلب فيها المعالج خانة ما.

2. المحلية الزمانية *Temporal Locality*: طلب نفس المعطيات بعد فترة زمنية قريبة.

مثال: وذلك كما في الحلقات *loops* فمثلاً عند وجود متحولات نعدل على قيمتها داخل الحلقات التكرارية كل مرة ندخل بها إلى الحلقات فنضع هذه المتحولات في الخابية وذلك لأننا سنطلب هذه المتحولات مرات عديدة وبأوقات قريبة.

أنواع التقابل في الخابيات

- ونقصد بالتقابلات بأنها طريقة وضع كتل الذاكرة RAM داخل الخابية وكيف يتم توضيع عناوين كتل الذاكرة الرئيسية داخل الخابية ولدينا ثلاثة أنواع للتقابلات في الخابية:

1. التقابل المباشر Direct Mapped:

تتوضع الكتلة في مكان محدد (وحيد) من الخابية أيّ كل كتلة لها مكان واحد لتتوضع فيه وعند البحث نبحث في مكان واحد فقط.

2. التجميعي التام Fully Associative

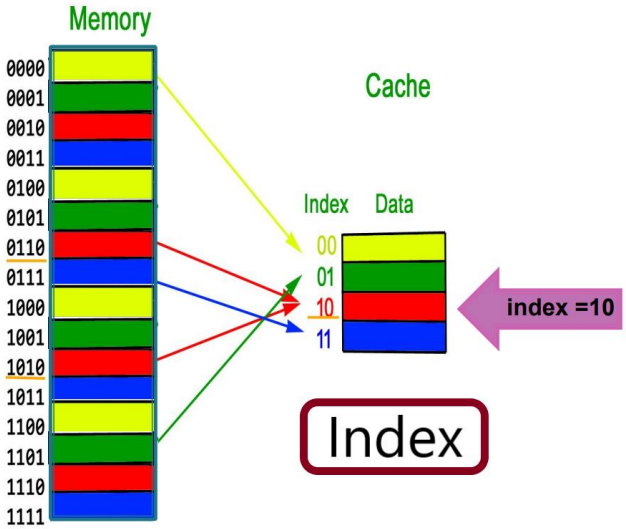
يمكن أن تتوضع الكتلة في أيّ مكان في الخابية وعند البحث عن كتلة ما يجب البحث في كامل الـ*cache* (الخابية).

3. التجميعي في مجموعات *set associative* (أهم نوع)

يمكن أن تتوضع الكتلة في أي مكان ضمن مجموعة محددة من الخابية أيّ هي الحل الوسطي بين المباشر والتام بحيث يقوم بتقسيم الخابية إلى مجموعات ويمكن أن تتوضع الكتلة في أيّ مكان ضمن المجموعة. وعملية البحث تكون ضمن المجموعة الواحدة وليس كامل الخابية.

تمرين 1

- بفرض خابية ذات تقابل مباشر $Direct\ Mapped$ وبفرض أن الذاكرة الخابية تتألف من 4 كتل (block).



1. حدد رقم سطر الخابية الذي ستتوضع في الكتلة ذات العنوان $0b1010$؟

- ملاحظة: يدل الرمز $(0b)$ أن العنوان ممثل بالنظام الثنائي أما في النظام السداسي عشر فالرمز هو $(0x)$

الحل

بما أن الذاكرة تقابل مباشر فلكل كتلة يوجد سطر وحيد إذ يوجد 4 أسطر في الـ$cache$.
يتم توزيع الكتل على الـindexes كما يلي:
الكتلة ذات العنوان 0 تقابل $index(0)$ في الـcache والكتلة ذات العنوان 1 تقابل الـ(1)index في cache وهكذا...
ولكن نحن ليس لدينا 4 كتل في RAM فقط كما في الرسم السابق نجد أن RAM مكونة من 16 سطر وكل كتلة هي سطر من سطور الـRAM ولذلك لنعرف الـindex المناسب لكتلة ما عنوانها بعد الـ4 نستخدم الـRAM القانون التالي:

$$I = J\ Mod\ C$$

حيث: I رقم سطر الخابية, J رقم الكتلة في الذاكرة, C عدد الأسطر في الخابية

$$1010 mod 100 = 0010 \Rightarrow 10\%4 = 2$$

أي تتوضع الكتلة المطلوبة في السطر $0b10$ في الخابية

2. من أجل نفس الخابية السابقة المؤلفة من 4 كتل, بفرض أن العنوان مؤلف من 4 بت, والكتلة مؤلفة من بايت واحد, حدّد عرض ومحتوى الحقول التي يتألف منها العنوان $0b1001$

الحل

كل عنوان في الذاكرة الرئيسية نقسمه إلى عدة حقول ليساعدنا على تحديد المكان المناسب لوضع الكتلة في الخابية, فيقسم العنوان الواحد إلى الحقول التالية:

tag	index	offset

- ترتيب هذه الحقول مهم جداً $tag - Index - offset$ ولسهولة تذكرها نجمع أول حرف من كل حقل فنحصل على كلمة (TIO)

Offset

لتحديد حجم حقل الـ$offset$ يجب أن ننظر إلى طريقة العنونة في الذاكرة (بالبايت/بالكلمة), العنونة في هذه السؤال بالكلمة حيث الكلمة مؤلفة من بايت واحد إذا ليس لدينا عدة بايتات في السطر الواحد للاختيار بينها, وبالتالي لا نحتاج إلى حقل الانزياح, ومنه لعنونة واحد بايت لا نحتاج إلى أيّ بت $(offset = 0bit)$ ولكن وبفرض أن الكلمة مؤلفة من بايتين $2\ byte$ هنا يجب وضع $offset = 1$ لعنونة حالتين (0 أو 1) وبفرض أن العنونة بالبايت حيث يجب لعنونة 4 حالات $\Leftarrow offset = 2$ وهنا نستنتج القانون التالي:

$$offset\ bits = \log_2(block\ size)$$

index

يستخدم الدليل index لتحديد رقم السطر في الخابية وكما في الطلب الأول تعاملنا مع الindex بأنه عدد أسطر الخابية حيث كان عدد الأسطر 4 (4 كتل) ولعنونة 4 حالات يلزمنا $(index = 2\ bit)$ وبالتالي لدينا القانون

$$index\ bits = \log_2(cache\ lines)$$

Tag

حقل الأمارة Tag يتألف من باقي بتات العنوان, وبناء على قيمته يتحدد هل الكلمة المطلوبة هي نفس الكلمة الموجودة في الخابية أم مختلفة عنها.

$$Tag\ bits = \text{Address} - (\text{index bits} + \text{offset bits})$$

$$\Rightarrow tag = 4 - 2 - 0 = 2\ bit$$

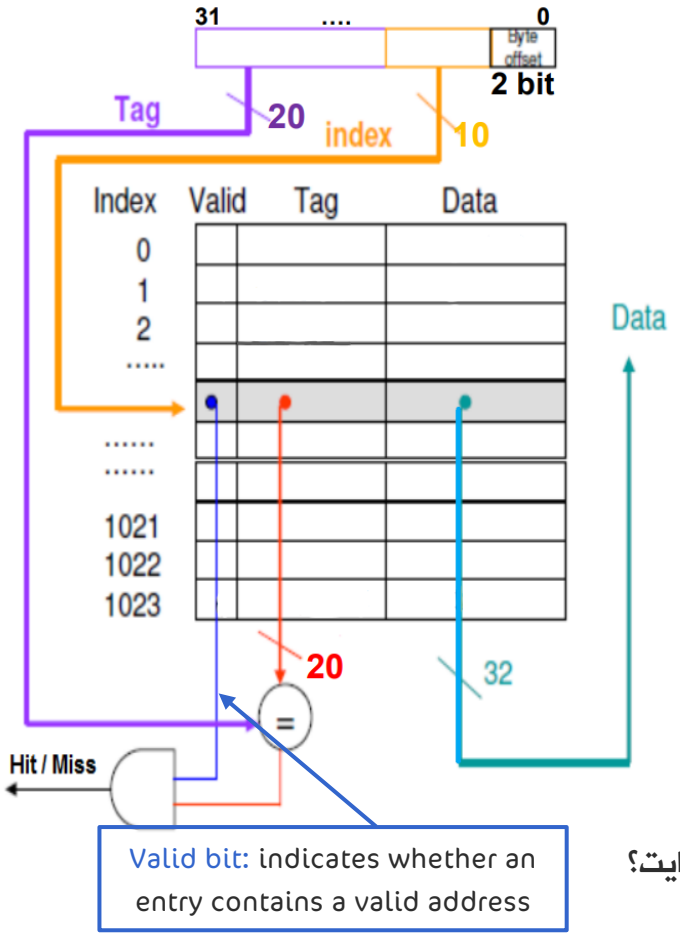
ومن أجل العنوان $0b1001$ فإن:

لا يوجد حقل انزياح

$$offset = 0, \quad tag = 10, \quad index = 01$$

تمرين 2

- بفرض كان لدينا الخابية الموضحة بالشكل التالي، أجب عن الأسئلة التالية:



1. ما نوع التقابل المستخدم؟

نلاحظ أنه من أجل كل قيمة index هناك tag وحيد يقابله فالخابية تقابل مباشر.

2. ما عدد بتات الحقل index؟

نلاحظ أن عدد أسطر (عدد الكتل) الخابية = 1024 (بدأنا العد من $0 \leftarrow 1023$)

فيكون $index = \log_2(1024) = 10bit$

3. ما عدد بتات الحقل $offset$ إذا علمت أن العنونة بالبايت؟

نلاحظ أن كتلة المعطيات مؤلفة من 32 بت $(0 \rightarrow 31)$ أيّ 4 بايت: بت $4 \times 8 = 32$ فيكون:

$$offset = \log_2 4 = 2\ bit$$

4. ما عدد بتات الحقل tag؟

$$tag = address - index - offset = 32 - 10 - 2$$
$$\Rightarrow tag = 20\ bit$$

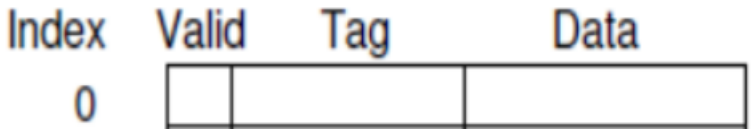
شرح الvalid

وهو الحقل الذي يدل على صلاحية أو عدم صلاحية البيانات في هذا السطر من الكاش (الخابية).

"It's okay to be discouraged. It's not okay to quit."

-Ryan Holiday

- نلاحظ من الرسمة السابقة أن السطر الواحد في الخابية $(cache)$ يتكون من عدة حقول:



- الحقل tag وهو نفسه حقل الأمارة الموجودة في عنوان في الذاكرة.
- الحقل $data$ ويحتوي على كتلة أو عدة كتل (حسب نوع التجميع) في الذاكرة RAM.
- بالإضافة إلى حقل الـvalid الذي شرحناه سابقاً.

بحيث عند إقلاع المعالج تأخذ جميع حقول الـdata في الأسطر قيماً عشوائية أو قيم صفرية وتأخذ جميع حقول valid القيم 0 لتدل على أن القيم الموجودة في هذه السطور ليس لها معنى وإنما هي عشوائية وعندما نأخذ block من الذاكرة RAM إلى سطر ما يتحول حقل الصلاحية (Valid) في هذا السطر إلى 1 ليدل على أن البيانات في هذا السطر أصبحت صالحة وهي ذات معنى.

الحقل $valid$
- 0 القيم غير صالحة
- 1 القيم الصالحة

ويساعدنا هذا الحقل في تحديد حالات الإصابة (H) أو الإخفاق (M) حيث يجب علينا مراعاة شرطين وهما:

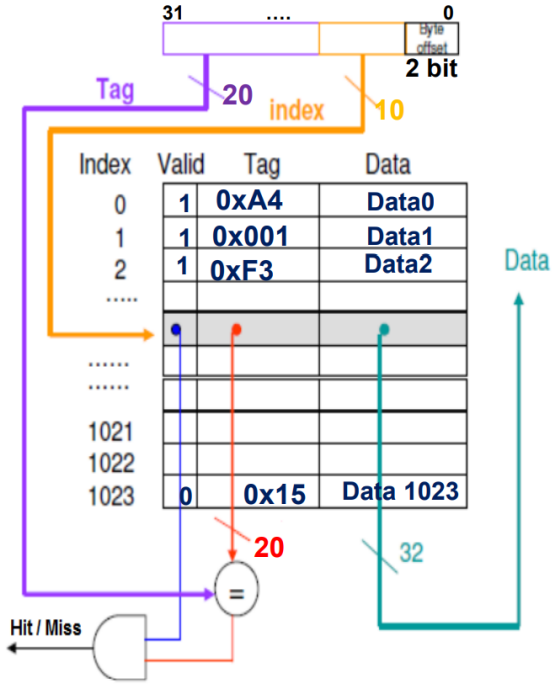
- قيمة الحقل $valid$ هو (1)
- قيمة tag المخزنة بالجدول مساوية لـtag العنوان.

أيّ في حال كان $valid = 1$ ← نذهب لنفحص الـtag المعرفة فيما إذا كانت إصابة أم إخفاق
أما إذا كان $valid = 0$ فوراً الحالة إخفاق $(Miss)$

تتمة التمرين (2)

5. حدد حالات الإصابة $H(Hit)$ والإخفاق $M(Miss)$ عند طلب العناوين التالية:

أولاً:



خطوات الحل

1. نقسم العنوان السابق إلى ثلاثة حقول TIO .
2. نبحث في الخابية عن السطر الذي يحمل الـ$index$ الذي حددناه.
3. نفحص حقل الصلاحية $V\ (valid)$ عند سطر الـ$index$:

0 ← فالحالة حتماً $M\ (miss)$ إخفاق.

1 ← نقارن الـtag الموجودة في العنوان مع الـtag الموجود في السطر فإذا كانا متساويان ← حالة إصابة Hit.

غير متساويان← حالة إخفاق $Miss$.

بالتالي سيكون لدينا كما حسبنا سابقاً:

$offset$ بت للإزاحة 2 ←

$index$ بت 10 ←

Tag بت للأمارة 20 ←

وبعد أن نقسم العنوان ونحدد الـindex سنجد أنه يساوي (2) أيّ $index = 2$.

ننظر إلى حقل الصلاحية في السطر $(index = 2)$ فنجد أنه valid = 1 أيّ صالح الآن سنقارن قيمة الـtag في العنوان مع tag السطر لنعرف فيما إذا كانت حالة إصابة أم إخفاق

قيمة الـtag في العنوان تساوي $0xF3$ وهي تساوي قيمة الـtag في السطر $(index = 2)$ ⇐حالة إصابة Hit

نحولها byte. في حال وجود قيمة بالـbitملاحظة: علينا أن ننتبه أننا نستخدم جميع القيم بالـ

ثانياً:

A 7

0000 0000 0000 1010 0111 0000 0000 0011

Tag index

سنحل بنفس الطريقة السابقة ولكن سنجد أن قيمتا الـtag مختلفتان ← فهي حالة إخفاق $M(miss)$.

ثالثاً

0000 0000 0000 0001 0101 1111 1111 1100

Tag index

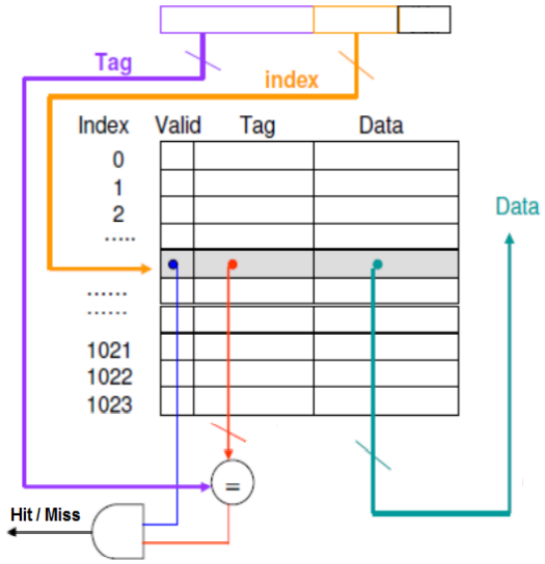
نلاحظ أن $v = 0$ في $index = 1023$ ⇐ حتماً حالة إخفاق $Miss$ ولا داعي لفحص قيمتي الـtag أصلاً.

6. تتمة التمرين 2: احسب الحجم الكلي للخابية.

- عرض السطر الواحد $= v + tag + block = 1 + 20 + 32$ ⇐ bit 53

الحجم الكلي للخابية = عرض السطر الواحد × عدد الأسطر $= 53 \times 1024 = 54272$ بت⇐ 6784 بايت

- ملاحظة الحجم الكلي للمعطيات $4k\ byte = 1k \times 4\ byte$ وهو يختلف عن الحجم الكلي للخابية.



7. ما عدد المقارنات المستخدمة؟

بما أن الخابية ذات تقابل مباشر فنحتاج إلى مقارن واحد فقط

للمقارنة بين قيمتي tag عند $idnex$ معين.

8. نفرض أن كتلة المعطيات مؤلفة من 8 كلمات والكلمة مؤلفة من 4 $byte$, وأن عدد أسطر الخابية 1024 والعنونة بالبايت, احسب عدد بتات الحقلين $offset$ و tag في هذه الحالة

$$Block = 8 \times 4 = 32\ Byte \Rightarrow offset = \log_2 32 = 5$$

$$\Rightarrow offset = 5$$

$$tag = 32 - 10 - 5 = 17\ bit \Rightarrow tag = 17\ bit$$

9. أعد الطلب السابقة ولكن بفرض العنونة بالكلمات $word\ addressed$

$$Block = 8\ word \Rightarrow offset = 3\ bit$$
$$\Rightarrow tag = 32 - 10 - 3$$
$$tag = 19\ bit$$

وظيفة

- بفرض لدينا خابية ذات تقابل مباشر مؤلفة من 4 كتل والكتلة من 4 بايت, والعنونة بالبايت وكانت فارغة عند البدء. حدد حالات الإصابة والإخفاق.

الحل:

Address (binary)	Hit/Miss ?
110001	
100111	
001111	
001100	
010001	
110010	
100101	
001110	
100001	
110101	

- بما أن التقابل مباشرة ← كل index له كتلة وحيدة.
- يوجد 4 كتل ← يوجد 4 أسطر في الخابية فتكون الخابية كما يلي:

Index	valid	Tag	Data
0			4 bytes
1			
2			
3			

1. لنبدأ بالعنوان (110001)

أولاً سنقسمه إلى 3 حقول وهي $TIO(Tag - index - offset)$

$$Index\ bits = \log_2(4) = 2\ bits$$
$$offset\ bits = \log_2(block\ size(bytes)) = \log_2 4 = 2\ bits$$
$$Tag = 6 - 2 - 2 = 2bits$$

وبما أن العنوان مؤلف من 6 بت بالتالي سيكون التقسيم بالشكل:

11	00	01
Tag	$index$	$offset$

ولكن بما أن الذاكرة فارغة فحتماً لا يمكننا أن نجد محتوى العنوان السابق في الخابية فتكون الحالة حتماً $Miss$

ونقوم بوضع محتويات العنوان السابق في السطر 00 فيكون الـTag $(Tag = 11)$

2. العنوان (100111)

10	01	11
Tag	$Index$	$offset$

ننظر إلى السطر ذو الـ$index$ 01 وبما أنه ما زال فارغ فالحالة هي حالة $Miss$ فيتم جلب الكتلة ذات العنوان

100111 إلى السطر الثاني ويصبح حقل الـTag $(Tag = 10)$.

3. العنوان (001111)

00	11	11
Tag	$Index$	$offset$

كما فعلنا سابقاً نذهب إلى السطر ذو الـ$index$ 11 أيّ 3 وبما أنه مازال فارغ أيضاً الحالة $Miss$ ويتم جلب الكتلة

ذات العنوان 001111 إلى السطر ذو الـ$index$ 11 ويكون حقل الـtag $(tag = 00)$

4. العنوان (001100)

00	11	00
Tag	$Index$	$offset$

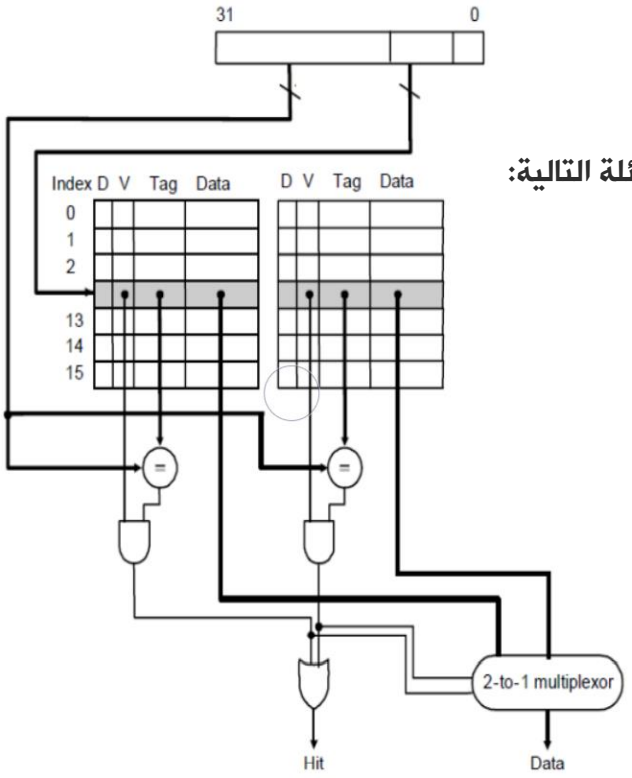
سنذهب للسطر ذو الـ$index$ 11 ونقارن قيمة الـTag في العنوان أيّ $(Tag = 00)$ مع قيمة الـTag في السطر

ذو الـ$index$ 11 أيّ $(Tag = 00)$ وبما أنهما متساويان ⇐ الحالة إصابة Hit.

ونكمل بنفس الطريقة وسيكون الحل كالتالي:

$$M \quad M \quad M \quad H \quad M \quad M \quad H \quad H \quad M \quad M$$

تمرين 3



- بفرض لدينا الخابية الموضحة بالشكل التالي, أجب عن الأسئلة التالية:

1. ما نوع التقابل المستخدم؟

نلاحظ أنه من أجل كل قيمة $index$ يوجد قيمتا tag عنده
فالخابية تجميعية في مجموعات $2-way-associative$
ونقصد هنا بالـ$2-way$ أن كل مجموعة set في الخابية تحوي سطرين (أو كتلتين) وليس عدد المجموعات.

وبالتالي ممكن أن يكون لدينا $3-way$ أو $4-way$ حسب عدد الأسطر في كل مجموعة.

2. ما عدد بتات الحقل $index$؟

نلاحظ أن عدد المجموعات في الخابية $= 16$, فيكون

$$index = \log_2(16) = 4bit$$

3. بفرض أن العنونة بالبايت، $offset = 3\ bit$ احسب حجم الكتلة

$$block = 2^{(offset)}$$
$$2^3 = 8\ byte$$

4. ما عدد بتات الحقل tag؟

$$Tag = 32 - 4 - 3 = 25\ bit$$
$$= @ - index - offset$$

5. احسب الحجم الكلي للخابية بفرض أن سياسة الكتابة خلفاً $write\ back$ ويوجد بت D $((Dirty))$

- ماذا نعني بسياسة الكتابة خلفاً $write\ back$؟

بما أننا نعلم أن ذاكرة $cache$ هي من وإلى الذاكرة RAM, فعند تعديل القيم على الـ$cache$ من قبل المعالج يجب أن نقوم بتحديث هذه القيمة أيضاً في الذاكرة الرئيسية ولتحقيق ذلك لدينا سياستين:

1. السياسة الأولى $(write\ through)$

وفي هذه السياسة يتم تحديث القيمة على الـ$cache$ ويتم أيضاً في نفس الوقت تحديث القيمة ذاتها في الذاكرة الرئيسية (هنا لا نحتاج لوجود بت D).

2. السياسة الثانية $(write\ back)$

في هذه السياسة يتم بشكل مبدأي تحديث القيمة في ذاكرة $cache$ فقط, ولكن عندما نريد الكتابة فوقه (استبدال) سطر ما في الـ$cache$ وكان هذا السطر يحتوي على قيمة محدثة ولم تتواجد بعد في الذاكرة ونقوم عندها بتحديث القيمة في الذاكرة الرئيسية ثم نقوم بحذف السطر السابق في الـ$cache$ والكتابة فوقه. وبذلك نحتاج إلى بت إضافي في سطر الـ$cache$ وهو بت $D\ (Dirty)$ والذي يأخذ القيمة 1 عندما يحتوي السطر في الكاش على قيمة محدثة لم تحدث بعد في الذاكرة الرئيسية ويأخذ القيمة (0) عندما لا يحتوي على بيانات محدثة عن الذاكرة الرئيسية, نلاحظ أنه في تعليمات الـsw أو sb (تعليمات التخزين في الذاكرة الرئيسية) ستكون قيمة الـd هي (1).

والآن بالعودة للطلب الخامس

عرض المجموعة الواحدة $2 \times (D + V + tag + block)$ ← سطرين $2\ way$

($D+V+tag+block$: للجدول الواحد)

$$2 \times (1 + 1 + 25 + 64) = 182\ bit$$

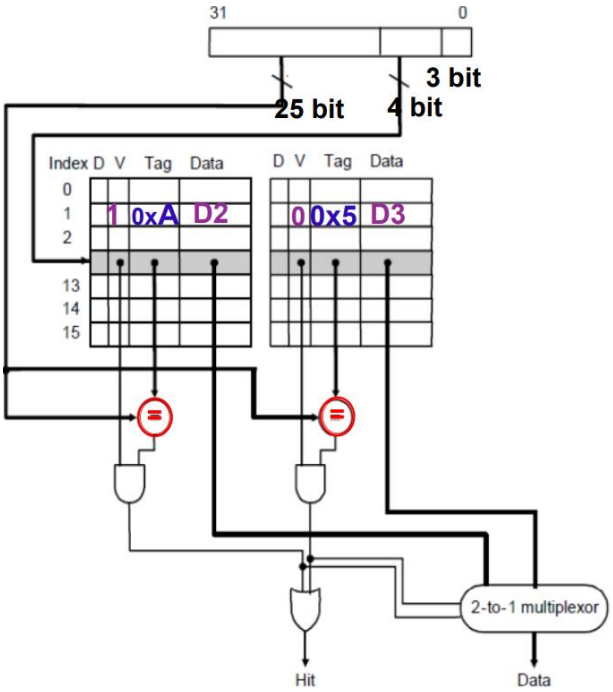
($block$: 2 في السطرين في المجموعة الواحدة, 16+16 عدد المجموعات ولدينا)

الحجم الكلي للخابية = عرض المجموعة الواحدة × عدد المجموعات ⇐

$$16 \times 182 = 2912\ bit$$

6. ما عدد المقارنات المستخدمة؟

بما أن الخابية $2 - way\ associative$ فنحتاج إلى مقارنتين



7. عند طلب العنوان التالي هل هي حالة إصابة أم إخفاق؟

$$0000\ 0000\ 0000\ 0111\ 1000\ 0\underbrace{000\ 1}_{index}011$$

نذهب إلى $(index = 1)$ وبما أن حقل الصلاحية في السطر الأول هو 1 وحقل الصلاحية في السطر الثاني هو 2 فسنقارن قيمتي الـtag فقط عند السطر الأول لدينا قيمة الـtag في العنوان المطلوب $0x8F0$ وقيمة الـtag في السطر الأول في $index = 1$ هي $0xA$ فهي غير متساويتان ⇐ حالة إخفاق $Miss$.

وظيفة

- بفرض لدينا خابية $4-way\ set\ associative$ وفيها حجم المعطيات الكلي $64\ Kbytes$، الكتلة مؤلفة من 16 كلمة، والكلمة مؤلفة من $32\ bit$، والعنونة بالكلمات، والعنوان من $40\ bit$ احسب:

1 - عدد بتات الحقل offset

2 - عدد بتات الحقل index

3 - عدد بتات الحقل tag

4 - عدد المقارنات المستخدمة

5 - حجم الخابية الكلي بفرض استخدام سياسة الكتابة أثناء التعديل write through (لا يوجد بت D)

1. بما أن العنونة بالكلمات نضع العنوان:

$$offset\ bit = \log_2(16) = 4bits$$

2. عدد بتات الحقل $index$

$$index\ bits = \log_2(number\ of\ cache\ lines).$$

ولحساب عدد سطور الـ$cache$:

عدد السطور = (حجم الـ$Data$ الكلي للـ$cache$)/(حجم كتل السطر الواحد بالبايتات)

حجم الـ$Data$ الكلي للـ$cache$= $64\ kbytes$

$$= (64 \times 1024bytes)$$

$$= 65536\ bytes$$

حجم كتل السطر الواحد في الـ$cache$ بالبايتات:

نعلم أن الكتلة مكونة من 16 كلمة وكل كلمة مؤلفة من 4 بايتات إذا الكتلة الواحدة تحتوي على 64 بايت ولكن السطر الواحد يحوي على 4 كتل لأن نوع التقابل هو $4-way\ associative$ إذاً يكون حجم الكتل في السطر الواحد: $64 \times 4 = 256$

ومنه يكون عدد الأسطر $\frac{65536}{256} = 256$

ومنه يكون

$$index\ bits = \log_2(256) = 8\ bits.$$

3. عدد بتات الحقل tag: بما أن العنوان مؤلف من $40\ bit$

$$\text{tag} = 40 - 8 - 4 = 28\ \text{bits}$$

4. عدد المقارنات المستخدمة يساوي إلى عدد الأمارات (tag) (أو الكتل) الموجودة في السطر الواحد ومنه نحتاج إلى 4 مقارنات.

5. الحجم الكلي للخابية = عرض السطر × عدد الأسطر

عرض السطر = $4 \times (v + tag + data\ block)$

$$= 4 \times (1 + 28 + 512)$$

$$= 541\ bits$$

ومنه: الحجم الكلي = $541 \times 1024 = 541\ kbits$

ملاحظات

- نقسم العنوان إلى ثلاثة حقول (الترتيب مهم جداً) $TIO: Tag\ index\ offset$
- نقسم السطر الواحد في الـ$cache$ إلى: $Valid\ Tag\ Data$
- التحويل من $Binary$ إلى $Hexadecimal$ (ثنائي إلى سداسي عشر):
نقوم بتجميع كل 4 بتات من اليمين في النظام الثنائي إلى بت واحد في النظام السداسي عشر وإعطاء الـ4 بتات القيمة المقابلة لهم في النظام السداسي عشر

مثال: $\underbrace{0000}_{0}\ \underbrace{0011}_{3}\ \underbrace{1111}_{F}$

- التحويل من $Hexadecimal$ إلى $Binary$ (سداسي عشر إلى ثنائي):
نقوم بتحويل كل بت في السداسي عشر إلى 4 بتات في الثنائي وتعطى لكل بت في السداسي عشر القيمة المقابلة لها بالثنائي

مثال: $\underbrace{0100}_{4}\ \underbrace{1010}_{A}\ \underbrace{0001}_{1}$

- يختلف الحجم الكلي للخابية عن الحجم الكلي للمعطيات في الخابية.
- قانون عدد المقارنات في حال التقابل المباشر والتقابل التجميعي في مجموعات:
عدد المقارنات = عدد الكتل في السطر الواحد
- قانون عدد المقارنات في نوع التقابل التجميعي التام:
عدد المقارنات = عدد الكتل الكلية في الـ$cache$

تمرين 4

- بفرض لدينا خابية تجميعية تامة fully associative مؤلفة من 4 كتل، الكتلة مؤلفة من كلمة واحدة،والكلمة من 4 بايت، والعنونة بالكلمات، ويتألف العنوان من 5 bit . احسب:

1 - عدد بتات الحقل

2- Offset عدد بتات الحقل

3 - Index عدد بتات الحقل

4 – Tag عدد المقارنات المستخدمة

5 - معدل الإصابة والإخفاق عند طلب تسلسل العناوين التالية 2, 5, 1, 2, 6, 5, 7,3 بفرض أن الخابية فارغة عند البدء.

- ثم احسب حجم الخابية الكلي، بفرض استخدام سياسة الكتابة خلفاً، وسياسة الاستعاضة LRU (يلزمها 2 بت لكل الخابية).

الحل:

1. بما أن العنونة بالكلمات فنضع القانون $offset\ bit = \log_2(1) = 0\ bit$
2. $offset = 0\ bit$ لأن الكتلة مؤلفة من كلمة واحدة فلا حاجة للإزاحة.
3. بما أن الخابية تجميعية تامة فإن الدليل غير موجود (يمكن وضع الكتلة في أي مكان في الخابية) أيّ $index = 0\ bits$

ملاحظة: عندما تكون الخابية تجميعية تامة دائماً يكون الـ0 = index

4. بما أن العنوان مؤلف من 5 bit نضع: Tag=5-0-0=5 bit

عدد المقارنات المستخدمة = عدد الأسطر في الخابية = 4 مقارنات

أيّ أن عدد المقارنات المستخدمة تساوي إلى عدد الكتل في الذاكرة لأنه سنقارن أمارة tag العنوان مع كل tag لكتل الـcache على التوازي وذلك لأنه لكل كتلة index كما في بقية التقابلات لذلك سنحتاج إلى 4 مقارنات.

ملاحظة: بما أن كلا الحقلين $(offset, index)$ غير موجودين في العنوان فالعنوان هو نفسه حقل الأمارة tag وبالتالي لا نحتاج إلى تقسيم العنوان لمعرفة الحقول المختلفة لأنه ليس لدينا إلا حقل واحد وهو tag وبما أن العنوان مكون من حقل واحد فيمكن للسهولة أن نعبر عن العنوان بالترميز العشري كما هو وارد في السؤال, وسنقوم بما كنا نقوم به سابقاً سنقارن أمارة العنوان مع أمارة السطر لنعرف ما إذا كانت حالة إصابة أو إخفاق ولكن بما أنه هنا أمارة العنوان هي نفسها العنوان فسنقارن العنوان كله مع الأمارة في سطر الكاش.

5. مناقشة حالة العنوان (2): بما أن الذاكرة كاش فارغة بدايةً فبالتأكيد هي حالة إخفاق $miss$ فيتم إحضار محتوى العنوان (2) إلى الذاكرة كاش ويتم وضعها مثلاً في السطر الأول منها (تذكر أنه في حالة التقابل التجميعي التام يمكن وضع أي كتلة في أي سطر ما أي أن الكتلة ليس لها مكان محدد)

مناقشة حالة العنوان (5): يتم مقارنة العنوان (5) مع الأمارة tag الوحيدة الموجودة في الذاكرة بالسطر الأول والتي تساوي 2 وبالتالي لأن الأمارتان مختلفتان نستنتج أن الحالة هي حالة إخفاق $miss$ ويتم إحضار محتوى العنوان (5) من الذاكرة RAM ووضعه في السطر الثاني من الكاش.

مناقشة حالة العنوان 1:

نقارن العنوان 1 مع كل الأمارات الموجود في الكاش (الأمارة 2 والأمارة 5) فنلاحظ أن كليهما لا يساويان العنوان وبالتالي هي حالة إخفاق $miss$ ويتم جلب محتوى العنوان 1 من الذاكرة RAM إلى الكاش ويتم وضعه في السطر الثالث.

مناقشة حالة العنوان 2:

نقارن العنوان 2 مع كل أمارة من الأمارات الموجود في الكاش $(1, 5, 2)$ نلاحظ أن السطر الأول له الأمارة ذاتها وبالتالي يوجد محتوى العنوان المطلوب في الكاش والحالة هي حالة إصابة hit.

مناقشة حالة العنوان 6:

نقارن العنوان 6 مع كل أمارة من الأمارات الموجود في الكاش $(1, 5, 2)$ نلاحظ أنه لا يوجد أمارة مساوي للعنوان المطلوب فهي حالة إخفاق $miss$ ويتم جلب محتوى العنوان 6 من الذاكرة الرئيسية ويتم وضعها في السطر الأخير من الكاش أيّ السطر الرابع.

مناقشة حالة العنوان 5:

نقارن العنوان 5 مع كل أمارة من الأمارات الموجود في الكاش $(6, 1, 5, 2)$ نلاحظ أن السطر الثاني له الأمارة ذاتها وبالتالي يوجد محتوى العنوان المطلوب في الكاش والحالة هي حالة إصابة hit.

مناقشة حالة العنوان 7:

نقارن العنوان 7 مع كل أمارة من الأمارات الموجود في الكاش $(6, 1, 5, 2)$ فنلاحظ أنه لا يوجد أمارة مساوية للعنوان المطلوب فنستبدله بأقدم قيمة لم نستخدمها ونلاحظ بأنها القيمة في السطر الثالث أيّ $Memory[1]$ فتصبح $Memory[7]$

مناقشة حالة العنوان 3:

نقارن العنوان 3 مع كل أمارة من الأمارات الموجود في الكاش $(6,7, 5, 2)$ فنلاحظ أنه لا يوجد أمارة مساوية للعنوان المطلوب فنستبدلها بأقدم قيمة لم نستخدمها ونلاحظ بأنها القيمة في السطر الأول أيّ القيمة $Memory[2]$ فتصبح $Memory[3]$

Address	Hit or miss	Contents of cache blocks after reference			
		line 0	line 1	line 2	line 3
2	Miss	Memory[2]			
5	Miss	Memory[2]	Memory[5]		
1	Miss	Memory[2]	Memory[5]	Memory[1]	
2	Hit	Memory[2]	Memory[5]	Memory[1]	
6	Miss	Memory[2]	Memory[5]	Memory[1]	Memory[6]
5	Hit	Memory[2]	Memory[5]	Memory[1]	Memory[6]
7	Miss	Memory[2]	Memory[5]	Memory[7]	Memory[6]
3	Miss	Memory[3]	Memory[5]	Memory[7]	Memory[6]

المحتوى النهائي للخابية

مما سبق نستنتج أنه لدينا: (2) حالة إصابة $2/8 = 25\% hit$

(6) حالة $6/8 = 75\%\ miss$

- حجم الخابية الكلي بفرض استخدام سياسة الكتابة خلفاً واستخدام الاستعاضة LRU يلزمها (2) بت لكل الخابية.

سياسة الاستعاضة LRU (Least Recently Used)

لنتعرف على سياسة الاستعاضة LRU وماذا تفعل ومتى تستخدم؟

- الذاكرة الخابية $cache$ في حال كونها ممتلئة كلياً ونريد إحضار كتلة إضافية إليها فتحتاج إلى إخلاء سطر أو أكثر منها لنقوم باستبدال الكتل الجديدة بسطور قديمة.
- الحجم الكلي للخابية: (عرض السطر × عدد السطور) + عدد البتات الإضافية اللازمة لسياسة الاستعاضة LRU. هنا أيضاً يتم استخدام سياسة الكتابة خلفاً وبالتالي نحتاج إلى البت D.

عرض السطر $= (V + D + Tag + Data\ Block)$

عرض السطر $= (1 + 1 + 5 + (4 \times 8)) = (1 + 1 + 5 + 32) = 39$

عدد السطور = عدد الكتل = 4

عدد البتات اللازمة لسياسة الاستعاضة = 2

وبالتالي $Total\ Cache\ Size = 4 \times 39 + 2 = 158\ bit$

وظيفة 3: أعد حل التمرين 5 بالحالتين:

الجواب: أ. تقابل مباشر:

مقارن واحد, $offset = 0\ bit\ ,\ index = 2\ bit\ ,tag = 3\ bit$

$MMMHMMM \quad Hit\ Rate = \frac{1}{7} = 14\% \quad Miss\ Rate = \frac{6}{7} = 86\% \quad Total\ cache\ size = 148\ bit$

ملاحظة: لا يلزم استخدام سياسة استعاضة LRU في التقابل المباشر.

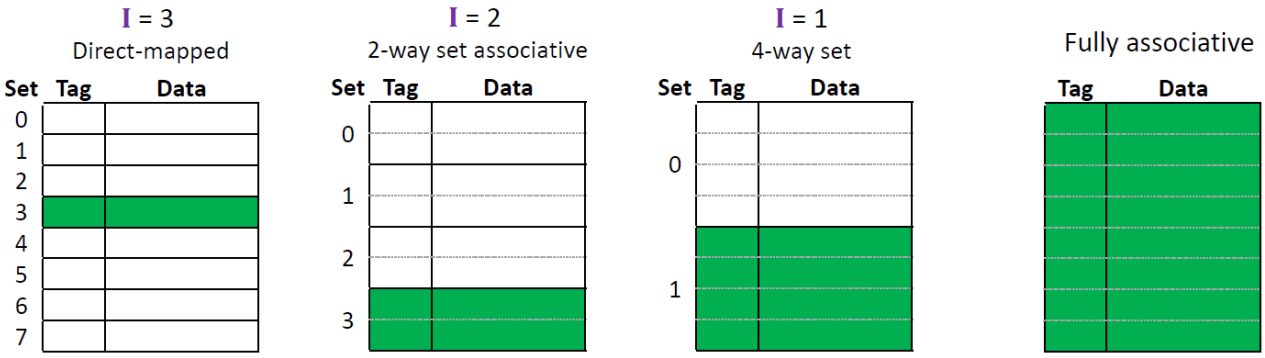
أ. تجميعية في مجموعات $2\ way\ associative$:

مقارنتين , $offset = 0\ bit\ ,\ index = 1\ bit\ ,tag = 4\ bit$

$MMMHMHH \quad Hit\ Rate = \frac{3}{7} = 43\% \quad Miss\ Rate = 57\% \quad Total\ cache\ size = 154\ bit$

وظيفة 4

- بفرض لدينا $block\ size = 16\ Byte, capacity = 8\ blocks, address = 16\ bits$
أين تتوضع المعطيات التي يتم طلبها من العنوان $0x1833$ في حالة خابية:



وظيفة 5

- بفرض لدينا خابية ذات تقابل مباشر direct mapped وفيها (1 byte) Data words are 8 bits,
word addressed, address is 20 bits, tag is 11 bits, Each block holds 16 bytes of data,
المطلوب حساب عدد الكتل في هذه الخابية.

الجواب: $32\ blocks$

وظيفة 6

بفرض لدينا خابية، العنونة بالبايت، والعنوان من 20 بت، سعة الخابية $64kB$ من المعطيات، وكتلة المعطيات مؤلفة من $32B$. المطلوب تحديد قيم الحقول T I O بفرض أن:

أ- الخابية ذات تقابل مباشر direct mapped . **ب- الخابية تجميعية تامة.**

الجواب: أ- $T = 4\ bit, I = 11\ bit, O = 5\ bit$ **ب-** $T = 15\ bit, I = 0\ bit, O = 5\ bit$

انتهت المحاضرة