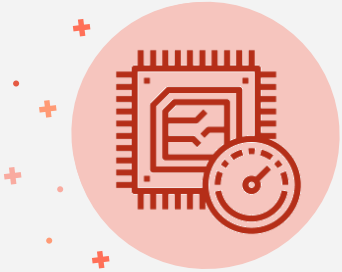


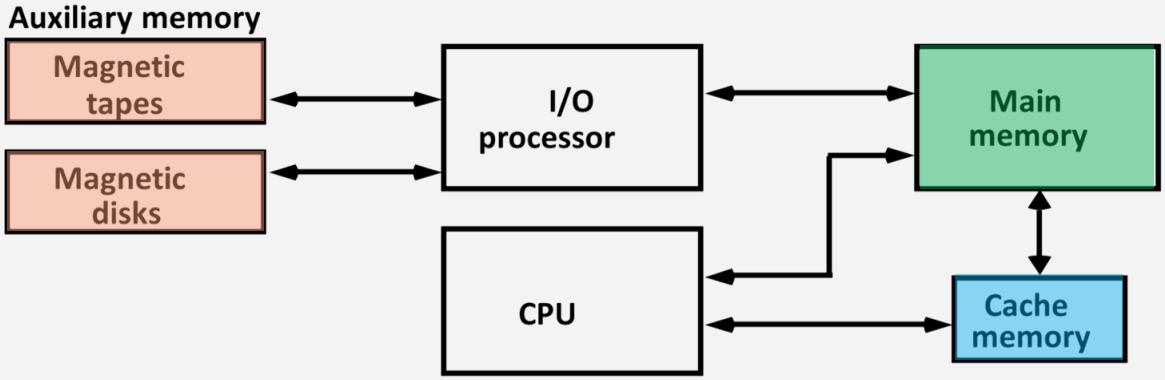
سیستم‌های دیجیتال ۲

اسلاید ۴

سازمان حافظه

محمدرضا رازقی‌زاده

سلسله مراتب حافظه

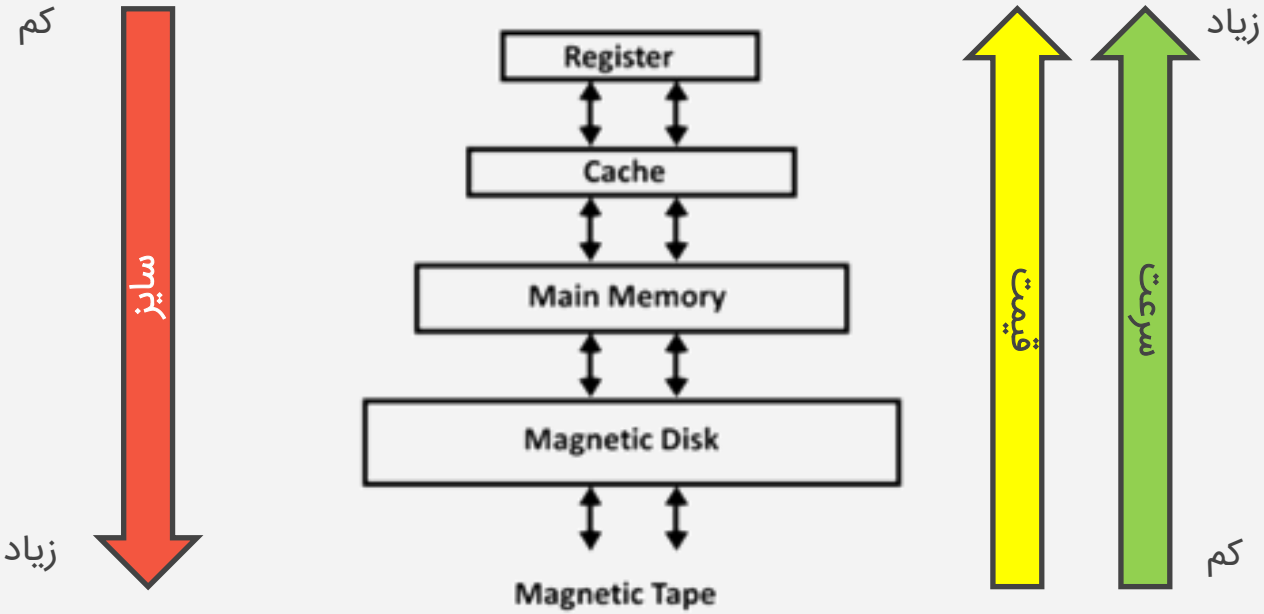


حافظه اصلی (Main Memory):
واحد حافظه‌ای که مستقیماً با CPU ارتباط برقرار می‌کند (RAM).

حافظه کمکی(Auxiliary Memory):
دستگاه‌هایی که ذخیره‌سازی پشتیبان را فراهم می‌کنند (Disk Drives).

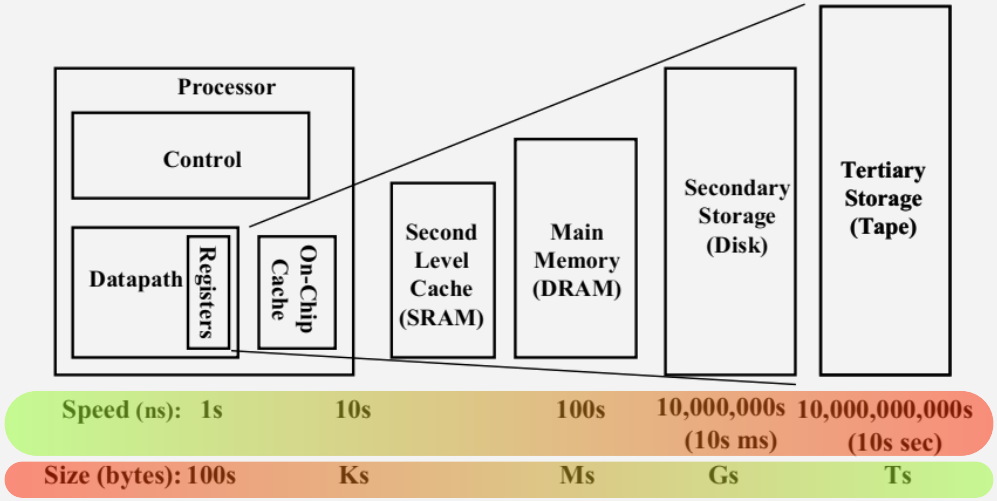
حافظه کش (Cache Memory):
حافظه‌ای بسیار پرسرعت که برای افزایش سرعت پردازش به کار می‌رود (Cache RAM).

سلسله مراتب حافظه



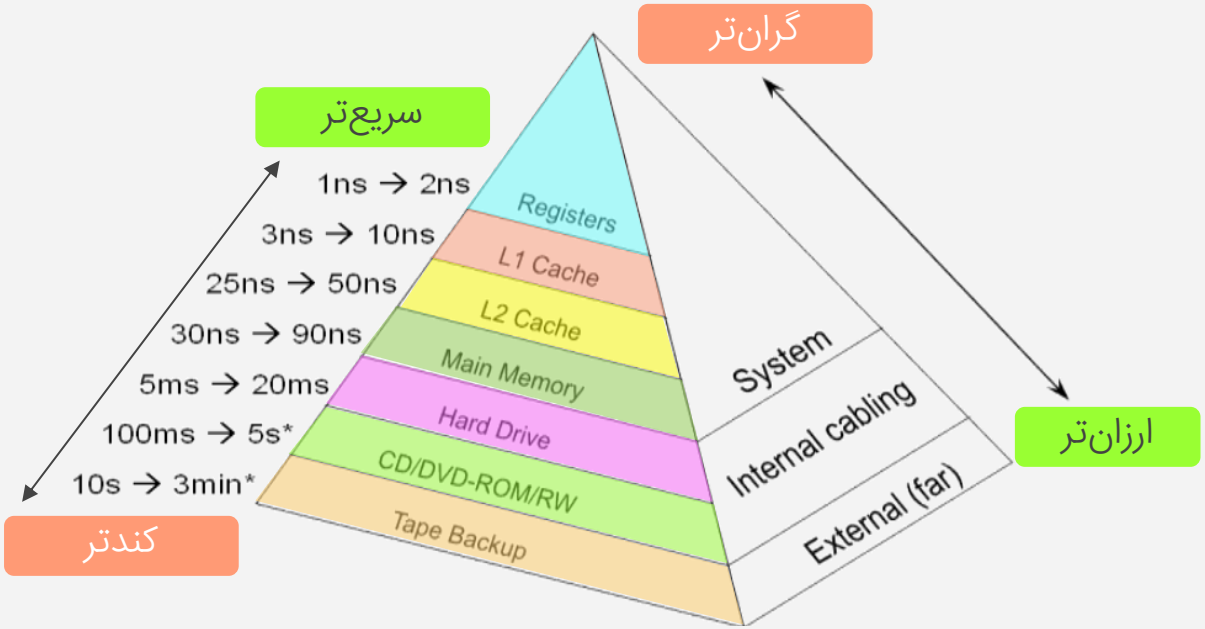
- سلسله مراتب حافظه با هدف سرعت بخشیدن دسترسی به حافظه با حداقل هزینه بوجود آمده است.

سلسله مراتب حافظه



- با حرکت از پردازنده به سمت حافظه‌های ثالثیه، سرعت کاهش و سایز افزایش می‌یابد.
- سیستم‌های مدرن از این سلسله‌مراتب برای بهینه‌سازی دسترسی به داده‌ها و کارایی کلی سیستم استفاده می‌کنند.

سلسله مراتب حافظه



- در سلسله مراتب حافظه اگر از پایین (کف هرم) به بالا حرکت کنیم:
  - ✓ سرعت دسترسی به حافظه بیشتر می شود.
  - ✓ هزینه سخت افزار افزایش پیدا می کند.
  - ✓ حجم حافظه کاهش پیدا می کند.

اهمیت حافظه

- حافظه از چه نظر اهمیت دارد؟

>> سرعت دسترسی به حافظه یك گلوگاه اصلی برای افزایش کارایی کامپیوتر است.

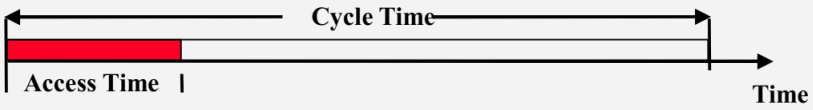
- در حالی که در سالیان گذشته حجم حافظه به سرعت افزایش پیدا کرده است، کاهش در زمان دسترسی به محتوای حافظه پیشرفت بسیار کمتری داشته است.

1000:1!

2:1!

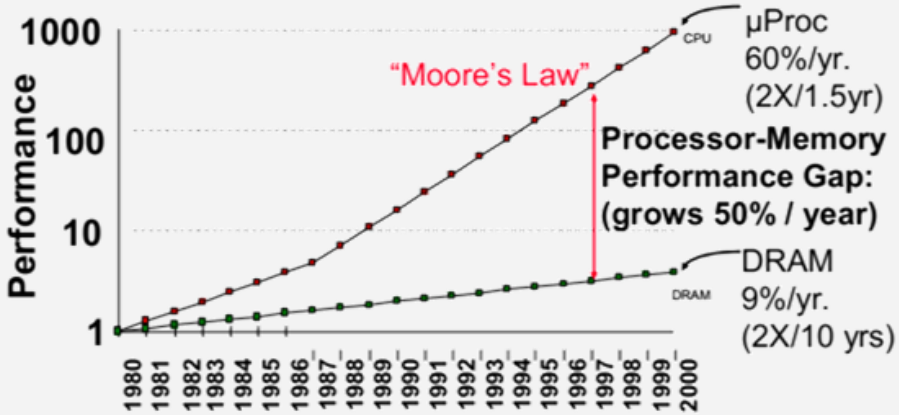
	DRAM	
Year	Size	Cycle Time
1980	64 Kb	250 ns
1983	256 Kb	220 ns
1986	1 Mb	190 ns
1989	4 Mb	165 ns
1992	16 Mb	145 ns
1995	64 Mb	120 ns

- زمان دسترسی به حافظه (Access Time) عبارت است از زمانی که بین تقاضای داده و آماده شدن آن طول می کشد.
- زمان سیکل (Cycle Time) عبارت است از زمان بین دو تکرار متوالی

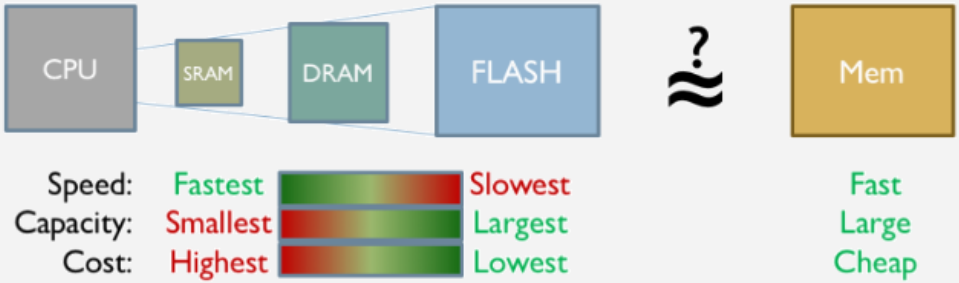


- برای مثال در حالی که زمان دسترسی به یك حافظه DRAM ممکن است ۶۰ ns باشد با احتساب زمان های لازم برای انتقال آدرس از پردازنده به حافظه، تاخیر باس ها، و ... زمان سیکل آن ممکن است به ۱۸۰-۲۵۰ ns برسد.

بر اساس قانون مور که گردون مور از بنیانگذاران شرکت اینتل آن را ارایه کرد، تعداد ترانزیستورهای روی یك تراشه (با مساحت ثابت) هر دو سال یك بار تقریبا دو برابر خواهد شد.

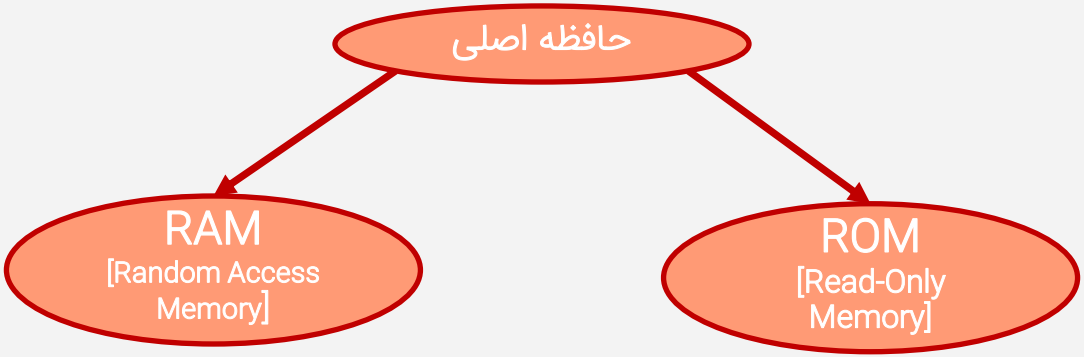


- **ویژگیهای حافظه:**
  - ✓ حافظه های بزرگ کند هستند
  - ✓ حافظه های سریع کوچك هستند
- **سؤال:** چگونه می توان حافظه بزرگ سریع و ارزانی داشت؟
  - ✓ استفاده از سلسله مراتب
  - ✓ دسترسی موازی



حافظه اصلی

- حافظه اصلی از مدارات سریعی ساخته می شود که برنامه ها و داده های مورد نیاز را در هنگام اجرا نگهداری می نماید. حافظه اصلی بر دو نوع است:



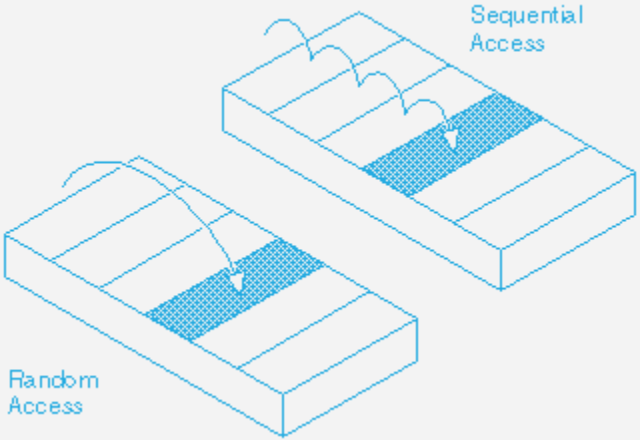
- از نظر دسترس‌پذیری تصادفی (Random Access)، هر دو حافظه ROM و RAM مشابه هستند، زیرا در هر دو نوع حافظه می‌توان به هر موقعیت حافظه به‌صورت مستقیم و تصادفی دسترسی پیدا کرد. به عبارت دیگر، در هر دو نوع حافظه، دسترسی به هر بخش از داده‌ها بدون نیاز به خواندن داده‌های پیشین انجام می‌شود.

- داده‌های با دسترسی تصادفی

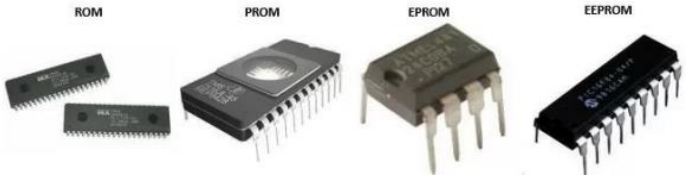
اجازه می‌دهند که به هر بخش از داده‌ها به‌طور مستقیم و مستقل دسترسی پیدا کنید (مانند RAM و HDD/SSD).

- داده‌های با دسترسی ترتیبی

نیازمند دسترسی به داده‌های قبلی برای رسیدن به بخش خاصی از داده‌ها هستند (مانند نوارهای مغناطیسی و نوار کاست).



- حافظه ROM حافظه‌ای است فقط خواندنی که محتوی آن یك بار نوشته شده و پس از نصب در کامپیوتر محتوای آن غیر قابل تغییر می‌باشد.
- معمولا از این حافظه برای ذخیره برنامه هایی نظیر bootstrap loader که برای راه اندازی اولیه کامپیوتر مورد نیاز هستند استفاده می شود.
- این حافظه انواع مختلفی دارد:

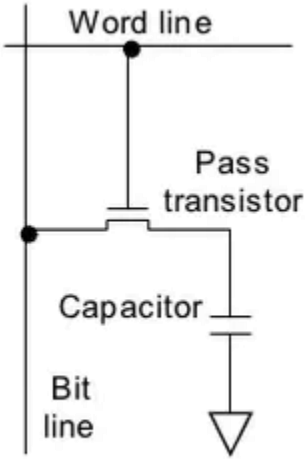


- PROM = Programmable ROM
- EPROM = Erasable Programmable ROM
- EEPROM = Electrical EPROM

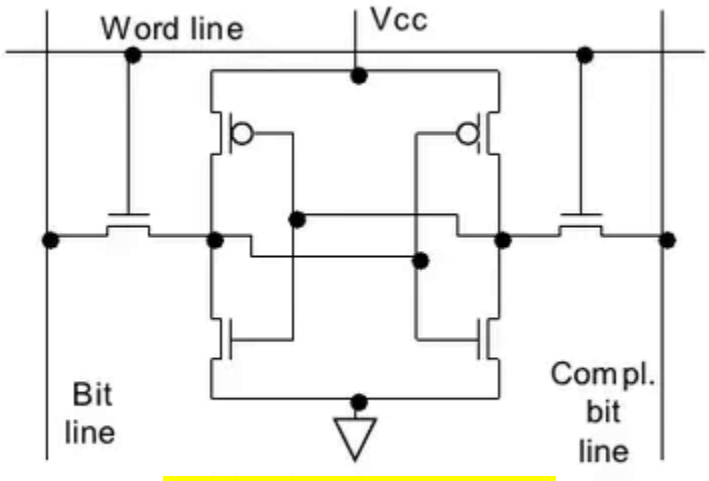
ROM

- عمده حافظه اصلی کامپیوتر از حافظه RAM ساخته می شود.
- معمولا در کامپیوترها دو نوع حافظه RAM مورد استفاده هستند:

- DRAM: Dynamic Random Access Memory
  - High density, low power, cheap, slow
  - Dynamic: need to be “refreshed” regularly
- SRAM: Static Random Access Memory
  - Low density, high power, expensive, fast
  - Static: content will last “forever” (until lose power)



ساختار حافظه DRAM

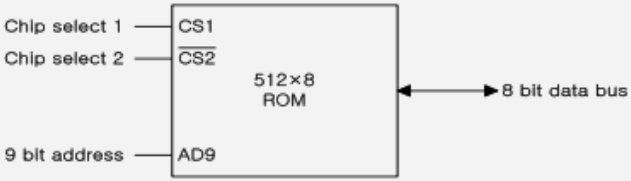


ساختار حافظه SRAM

ویژگی	SRAM (Static RAM)	DRAM (Dynamic RAM)
مخفف	Static Random Access Memory	Dynamic Random Access Memory
ساختار ذخیره‌سازی	از فلیپ-فلاپ‌ها و ترانزیستور برای ذخیره هر بیت استفاده می‌کند.	از خازن و ترانزیستور برای ذخیره هر بیت استفاده می‌کند.
نیاز به تازه‌سازی	نیاز به تازه‌سازی (Refresh) ندارد.	نیاز به تازه‌سازی مداوم دارد تا داده‌ها حفظ شوند.
سرعت	سریع‌تر	کندتر در مقایسه با SRAM
تراکم حافظه	تراکم کمتر (نیاز به فضای بیشتری برای هر بیت).	تراکم بالاتر (نیاز به فضای کمتر برای هر بیت)
مصرف توان	مصرف توان کمتر در حالت ثابت، اما در حالت فعال بیشتر است.	مصرف توان کمتر در حالت فعال، اما به دلیل تازه‌سازی مداوم، مصرف توان کلی بیشتر است.
پیچیدگی طراحی	طراحی پیچیده‌تر به دلیل تعداد بیشتر ترانزیستورها.	طراحی ساده‌تر به دلیل استفاده از خازن‌ها.
ظرفیت حافظه	معمولاً ظرفیت‌های کوچک‌تر (چند مگابایت)	معمولاً ظرفیت‌های بزرگ‌تر (چند گیگابایت)
هزینه	گران‌تر به دلیل پیچیدگی و تعداد ترانزیستورها.	ارزان‌تر به دلیل تراکم بالا و طراحی ساده‌تر.
کاربردها	-کش CPU (L۱، L۲، L۳) - حافظه‌های با سرعت بالا	-حافظه اصلی (RAM) در کامپیوترها و سرورها - کارت‌های گرافیکی
زمان دسترسی	کوتاه‌تر (چند نانوثانیه)	بلندتر (چند ده نانوثانیه)
پایداری داده	تا زمانی که تغذیه برقرار باشد، پایدار است.	نیاز به شارژ مداوم خازن‌ها برای حفظ داده‌ها.

بلوك دیاگرام حافظه

- ✓ Typical RAM chip
  > 128 X 8 RAM : 27 = 128 (7 bit address lines)
- ✓ Typical ROM chip
  > 512 X 8 ROM : 29 = 512 (9 bit address lines)



CS ۱	CS ۲	R D	W R	عمل حافظه	وضعیت گذرگاه داده
۰	۰	X	X	مسدود	امپدانس بالا
۰	۱	X	X	مسدود	امپدانس بالا
۰	۰	۰	۱	مسدود	امپدانس بالا
۱	۰	۰	۱	نوشتن	ورودی داده به RAM
۱	۰	۱	۰	خواندن	خروجی داده به RAM
۱	۱	X	X	مسدود	امپدانس بالا

- در زمان طراحی کامپیوتر مقدار حافظه اختصاص داده شده به هر یك از حافظه های RAM , ROM مشخص می گردند.به نحوه آدرس دهی حافظه‌ها در کنار یکدیگر نقشه آدرس حافظه گفته می‌شود.

مثال: فرض کنید کامپیوتری با مشخصات زیر در اختیار داریم:

Memory Configuration : **512** bytes RAM + **512** bytes ROM

» 4 x 128 bytes RAM + 1 x 512 byte ROM

Component	Address	A_{10} A_9 A_8 A_7 A_6 A_5 A_4 A_3 A_2 A_1
RAM1	0000-007F	0 0 0 x x x x x x x
RAM2	0080-00FF	0 0 1 x x x x x x x
RAM3	0100-017F	0 1 0 x x x x x x x
RAM4	0180-01FF	0 1 1 x x x x x x x
ROM	0200-03FF	1 x x x x x x x x x

Memory Configuration : **512** bytes RAM + **512** bytes ROM

» 4 x 128 bytes RAM + 1 x 512 byte ROM

Memory Address Map

» Address line 9 8

- RAM 1 0 0 : 0000 - 007F
- RAM 2 0 1 : 0080 - 00FF
- RAM 3 1 0 : 0100 - 017F
- RAM 4 1 1 : 0180 - 01FF

» Address line 10

- ROM 1 : 0200 - 03FF

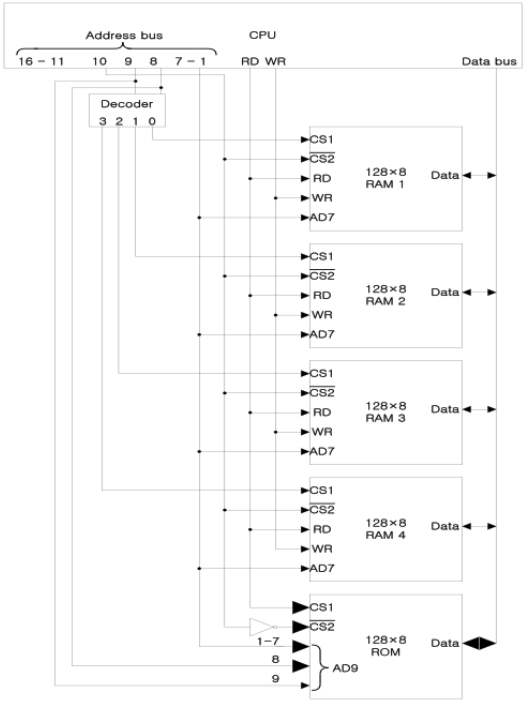
Memory Connection to CPU

» 2 x 4 Decoder : RAM select (CS1)

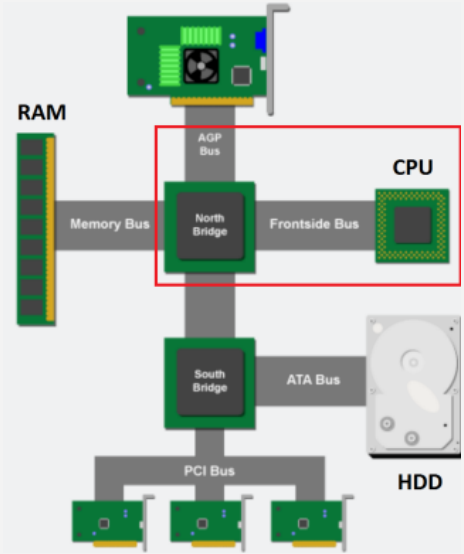
» Address line 10

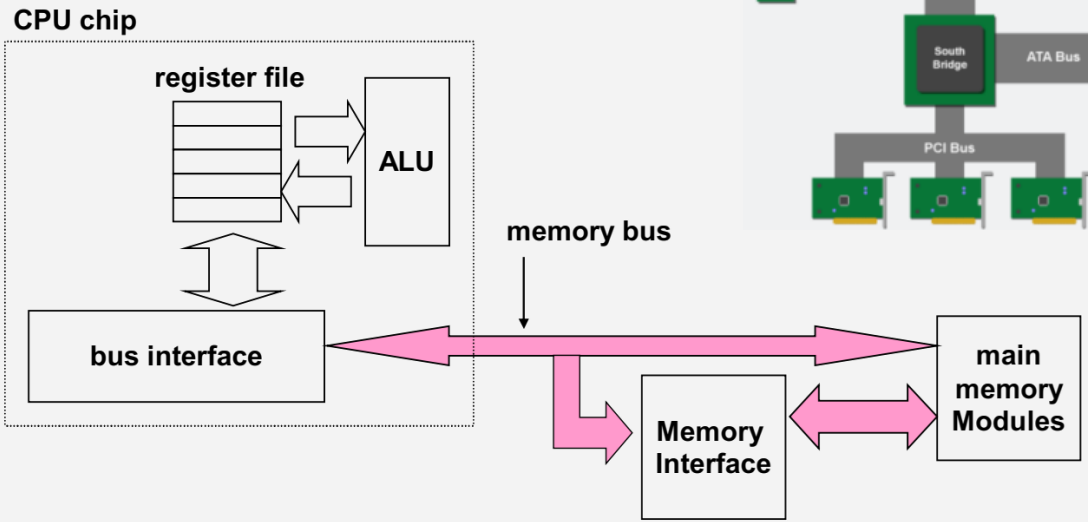
- RAM select : CS2
- ROM select : CS2 Invert

Component	Address	A_{10} A_9 A_8 A_7 A_6 A_5 A_4 A_3 A_2 A_1
RAM1	0000-007F	0 0 0 x x x x x x x
RAM2	0080-00FF	0 0 1 x x x x x x x
RAM3	0100-017F	0 1 0 x x x x x x x
RAM4	0180-01FF	0 1 1 x x x x x x x
ROM	0200-03FF	1 x x x x x x x x x



ارتباط حافظه اصلی با پردازنده

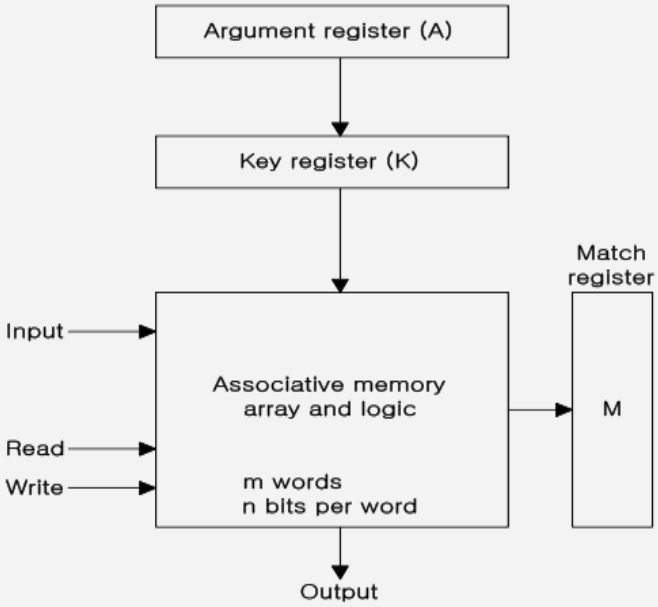




حافظه شرکت پذیر یا هم پیوند Associative Memory (AM)

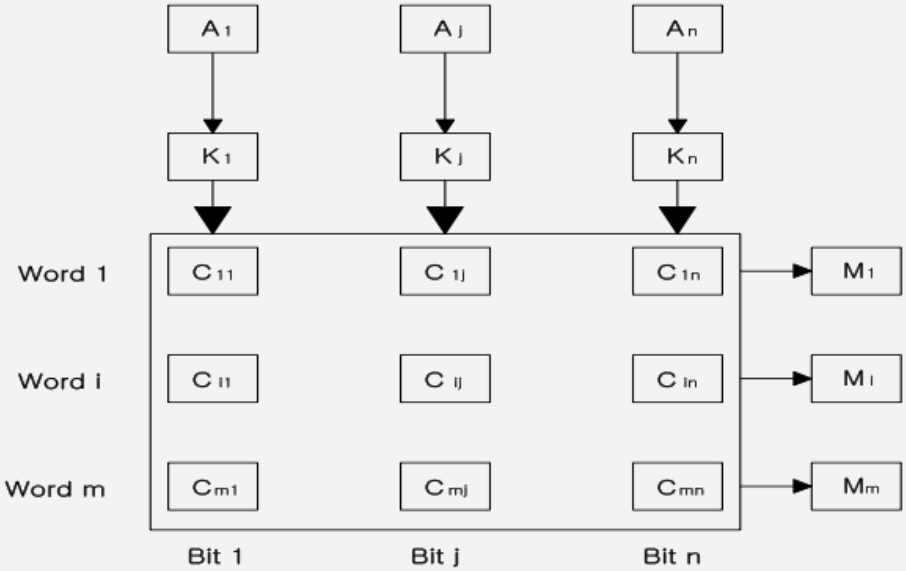
- از این حافظه برای تسریع عمل جستجو در بین داده های ذخیره شده استفاده می‌شود.
- برای کاهش زمان جستجو به جای استفاده از آدرس، از محتوی خود داده استفاده می شود.
- نام دیگر این روش Content Addressable Memory (CAM) می‌باشد.
- هنگام نوشتن داده در این حافظه نیازی به آدرس نیست!
- این حافظه قادر است تا جای خالی را پیدا نموده و داده را در آنجا ذخیره می نماید.
- هنگام خواندن داده از حافظه، مقدار داده و یا بخشی از آن به حافظه ارائه شده و حافظه تمامی کلمات ذخیره شده ای را که با داده مورد نظر یکسان هستند را علامت گذاری کرده و آماده خواندن می نماید.

بلوک دیاگرام حافظه شرکت پذیر



- هر کلمه در حافظه به طور موازي با محتويات ثبات (Argument Register) A مقايسه مي شود.
- به کمک ثبات کليد (Key Register) مي توان محدوده جستجو را کاهش داد.
- اگر در حافظه، word[i]=A آنگاه M[i]=۱. در غير اين صورت M[i]=۰.
- تمام کلماتي که بيت متناظر آنها در M يك شده باشد، از حافظه خوانده مي شوند.

عملکرد حافظه شرکت پذیر



عملکرد حافظه شرکت پذیر

مثال ۱

									M
A	1	1	0	1	1	0	0	1	
K	1	1	1	1	1	1	1	1	
Word 0	1	1	1	0	0	0	1	1	0
Word 1	1	1	0	0	0	0	0	1	0
Word 2	1	0	1	0	1	0	1	0	0
Word 3	0	1	0	1	0	1	0	1	0
Word 4	1	1	1	1	1	1	1	1	0
Word 5	1	1	0	1	1	0	0	1	1
Word 6	0	0	0	0	1	1	1	0	0
Word 7	1	1	1	1	1	1	1	1	0

عملکرد حافظه شرکت پذیر

مثال ۲

									M
A	1	1	0	1	1	0	0	1	
K	1	1	1	0	0	1	1	0	
Word 0	1	1	1	0	0	0	1	1	0
Word 1	1	1	0	0	0	0	0	1	1
Word 2	1	0	1	0	1	0	1	0	0
Word 3	0	1	0	1	0	1	0	1	0
Word 4	1	1	1	1	1	1	1	1	0
Word 5	1	1	0	1	1	0	0	1	1
Word 6	0	0	0	0	1	1	1	0	0
Word 7	1	1	1	1	1	1	1	1	0

حافظه نهان (Cache)

- اگر قسمت فعال برنامه و داده ها در حافظه سريع و كوچكي قرار داده شود، مي توان با كم كردن ميانگين زمان دسترسي به حافظه، زمان اجراي برنامه را كاهش داد.
- اين حافظه سريع و كوچك را حافظه cache مي نامند.
- معمولا حافظه cache زمان دسترسي بسيار كمتري نسبت به حافظه اصلي دارد (۵ تا ۱۰ برابر كمتر)
- دريك كامپيوتر ممكن است عمل cache تا چند سطح تكرار گردد.

حافظه نهان (Cache)

- وقتي كه CPU نياز به دسترسي به حافظه دارد ابتدا حافظه cache را جستجو مي نمايد؛ اگر داده در اين حافظه سريع موجود بود (hit) از آن استفاده مي شود در غير اينصورت (miss) با رجوع به حافظه، بلوكي از داده‌ها كه شامل داده مورد نياز CPU مي شود از حافظه اصلي به حافظه cache منتقل مي گردد.

- ممكن است CPU منتظر نوشته شدن داده و كلمات مجار آن (يك بلوك) در حافظه نهان نماند و داده را مستقيما از حافظه اصلي دريافت كند.

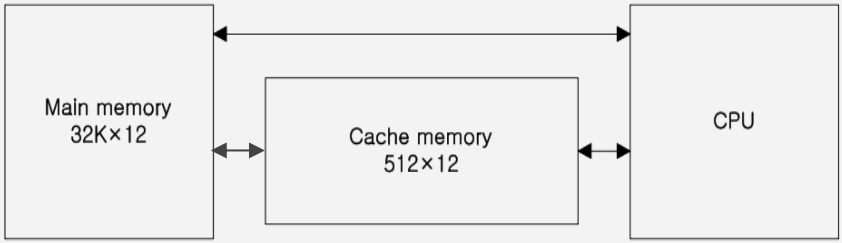
درصد موفقیت

- کارايي حافظه cache با ضريبي با نام درصد موفقيت (نرخ دسترسي) hit ratio اندازه‌گيري مي شود.
- اين درصد عبارت است از نسبت تعداد دفعاتي که داده در cache يافت شده به تعداد کل دفعات رجوع به حافظه.
- هر چه مقدار اين درصد بيشتر باشد سرعت دسترسي به حافظه به سرعت cache نزديکتر مي‌شود.

$$\text{درصد موفقیت (h)} = \frac{\text{تعداد hit ها}}{\text{تعداد کل مراجعات}} = \frac{\text{تعداد hit ها}}{\text{تعداد hit ها + تعداد miss ها}}$$

زمان متوسط دسترسي

فرض: حافظه اصلي مي تواند مستقيما داده در اختيار CPU قرار دهد.



teff : مدت زمان دستيابي متوسط (موثر) cache

t_{cache} : مدت زمان دستيابي cache

t_{main} : مدت زمان دستيابي حافظه اصلي

h : درصد موفقيت

$$t_{eff} = h \cdot t_{cache} + (1-h) \cdot t_{main}$$

زمان متوسط دسترسي

مثال: فرض كنيد يك كامپيوتر با مشخصات زير در اختيار داريم:

- ✓ زمان دسترسي ۱۰۰ ns براي حافظه cache
- ✓ زمان دسترسي ۱۰۰۰ ns براي حافظه اصلي
- ✓ درصد موفقيت ۰.۹
- ✓ حافظه اصلي مي تواند مستقيما به CPUداده بدهد

زمان متوسط دسترسی را محاسبه نماييد؟

$t_{cache} = 100\ ns$

$t_{main} = 1000ns$

$h = 0.9$

$t_{eff} = h \cdot t_{cache} + (1-h) \cdot t_{main}$

$t_{eff} = (0.9) \cdot (100ns) + (0.1) \cdot (1000ns) = 190\ ns$

زمان متوسط دسترسي

فرض: حافظه اصلي نمي تواند مستقيما داده در اختيار CPU قرار دهد.

teff : مدت زمان دستيابي متوسط (موثر) cache

t_{cache} : مدت زمان دستيابي cache

t_{main} : مدت زمان دستيابي حافظه اصلي

h : درصد موفقيت

$$t_{eff} = h \cdot t_{cache} + (1-h) \cdot (t_{main} + t_{cache}) = \cdot t_{cache} + (1-h) \cdot t_{main}$$

زمان متوسط دسترسي

مثال: فرض كنيد يك كامپيوتر با مشخصات زير در اختيار داريم:

- ✓ زمان دسترسي ۱۰۰ ns براي حافظه cache
- ✓ زمان دسترسي ۱۰۰۰ ns براي حافظه اصلي
- ✓ درصد موفقيت ۰.۹
- ✓ حافظه اصلي مي تواند مستقيما به CPUداده ندهد

زمان متوسط دسترسی را محاسبه نماييد؟

$t_{cache} = 100\ ns$

$t_{main} = 1000ns$

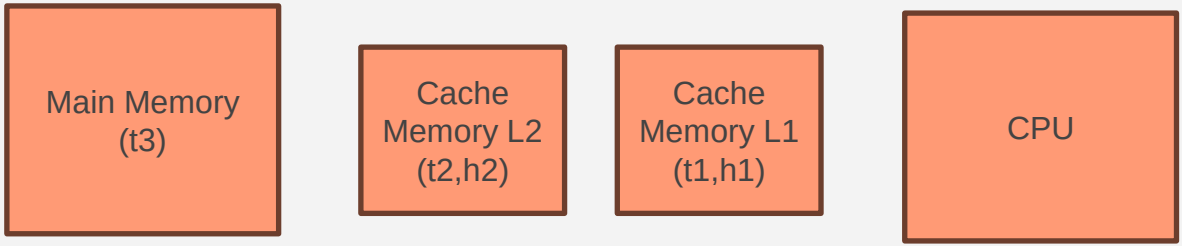
$h = 0.9$

$$t_{eff} = t_{cache} + (1-h) \cdot t_{main}$$

$$t_{eff} = (100ns) + (0.1) \cdot (1000ns) = 200\ ns$$

زمان متوسط دسترسي

زمان متوسط دسترسی برای یک سیستم با چندین سطح cache به صورت زیر محاسبه می گردد:



اگر CPU مستقیما به حافظه متصل نباشد:

$$t_{eff} = t_1 + (1-h_1) \cdot t_2 + (1-h_1) \cdot (1-h_2) \cdot t_3$$

اگر CPU مستقیما به حافظه متصل باشد:

$$t_{eff} = h_1 \cdot t_1 + (1-h_1) h_2 \cdot t_2 + (1-h_1) \cdot (1-h_2) \cdot t_3$$

تاثیر اندازه بلوك هاي cache

در حافظه کش، اندازه بلوک‌ها نقش مهمی در عملکرد آن دارد. اگر اندازه بلوک‌ها بزرگ‌تر باشد:

بهره‌مندی از محلی‌سازی فضایی Spatial Locality:

داده‌هایی که نزدیک به هم در حافظه اصلی قرار دارند، احتمالاً با هم نیاز خواهند شد. بلوک‌های بزرگ‌تر می‌توانند داده‌های بیشتری را یکجا وارد کش کنند و این باعث کاهش نیاز به دسترسی مکرر به حافظه اصلی می‌شود.

افزایش جریمه خطا Miss Penalty:

زمانی که داده مورد نظر در کش وجود ندارد، نیاز به انتقال یک بلوک کامل از حافظه اصلی به کش است. هرچه بلوک بزرگ‌تر باشد، زمان بیشتری برای این انتقال نیاز خواهد بود و جریمه خطا افزایش می‌یابد.

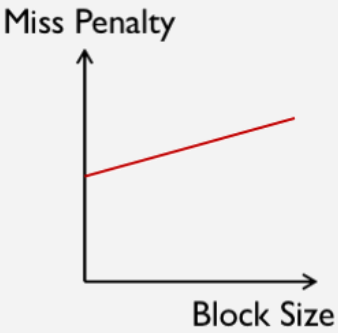
تأثیر بر زمان و نرخ دسترسی:

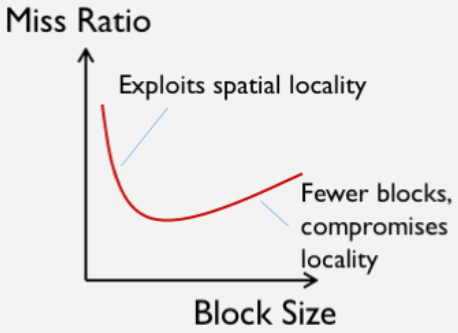
بلوک‌های بزرگ‌تر ممکن است زمان دسترسی به کش Hit Time را افزایش دهند و همچنین به دلیل پر شدن سریع‌تر کش، نرخ خطا Miss Rate نیز بالا برود.بنابراین، زمان دسترسی میانگین AMAT با توجه به این فاکتورها محاسبه می‌شود:

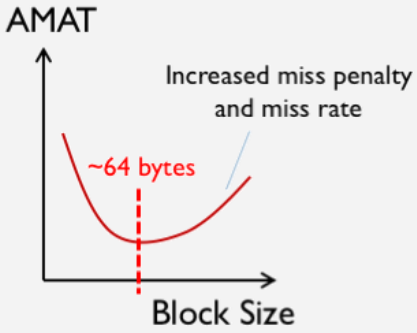
$$AMAT = \text{Hit Time} \times \text{Hit Ratio} + \text{Miss Penalty} \times (1 - \text{Hit Ratio})$$

تاثیر اندازه بلوك هاي cache

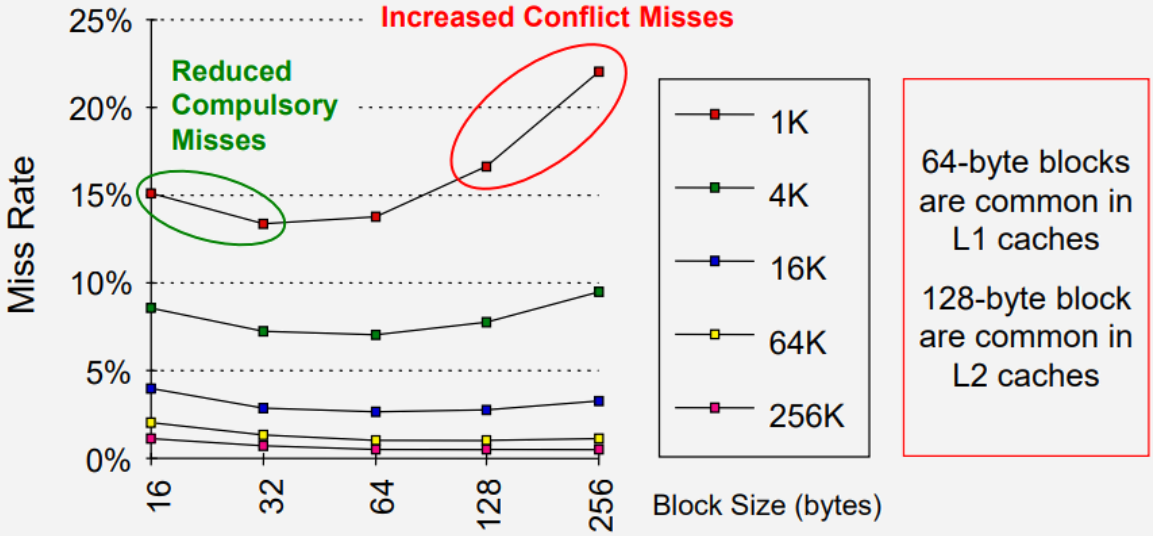
به بیان ساده، بلوک‌های بزرگ‌تر می‌توانند عملکرد را بهبود دهند اما اگر بیش از حد بزرگ باشند، ممکن است اثر معکوس بگذارند و کش کندتر عمل کند.



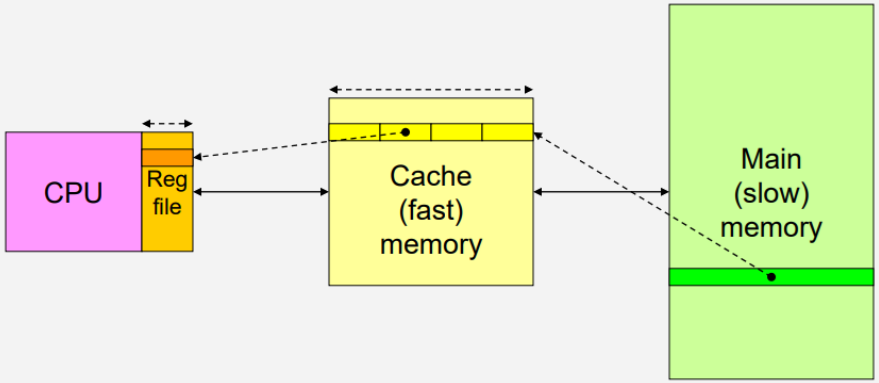




تاثير اندازه بلوك هاي cache



نگاشت



عمل انتقال داده از حافظه اصلي به حافظه cache نگاشت نامیده می‌شود. این کار به سه طریق قابل انجام است:

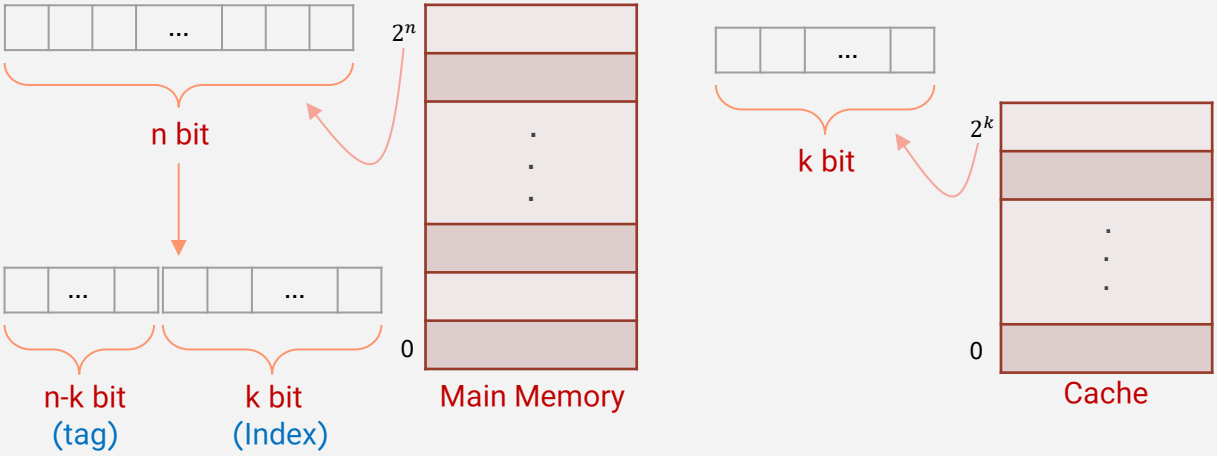
نگاشت مستقیم (Direct Mapping)

نگاشت شرکت پذیر (Associate Mapping)

نگاشت مجموعه هاي هم پیوند (Set-associative mapping)

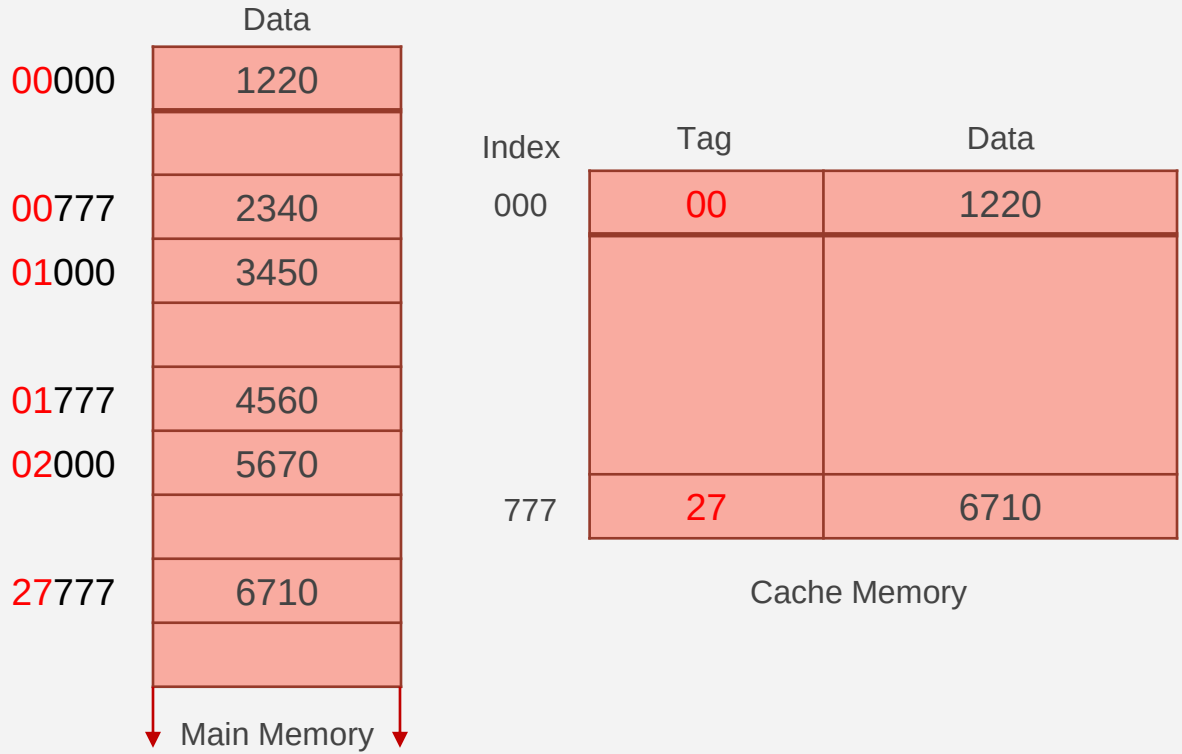
نگاشت مستقیم (Direct Mapping)

- در این روش براي پياده سازي حافظه کش از يك حافظه RAM سريع استفاده می‌شود:
- اگر حافظه اصلي داراي 2^n کلمه و حافظه کش داراي 2^k کلمه باشند، در نتيجه به ترتيب برای هريک به n و k بيت آدرس نياز خواهد بود.
- در اين صورت آدرس n بيتي حافظه اصلي به صورت زير تقسيم مي شود:



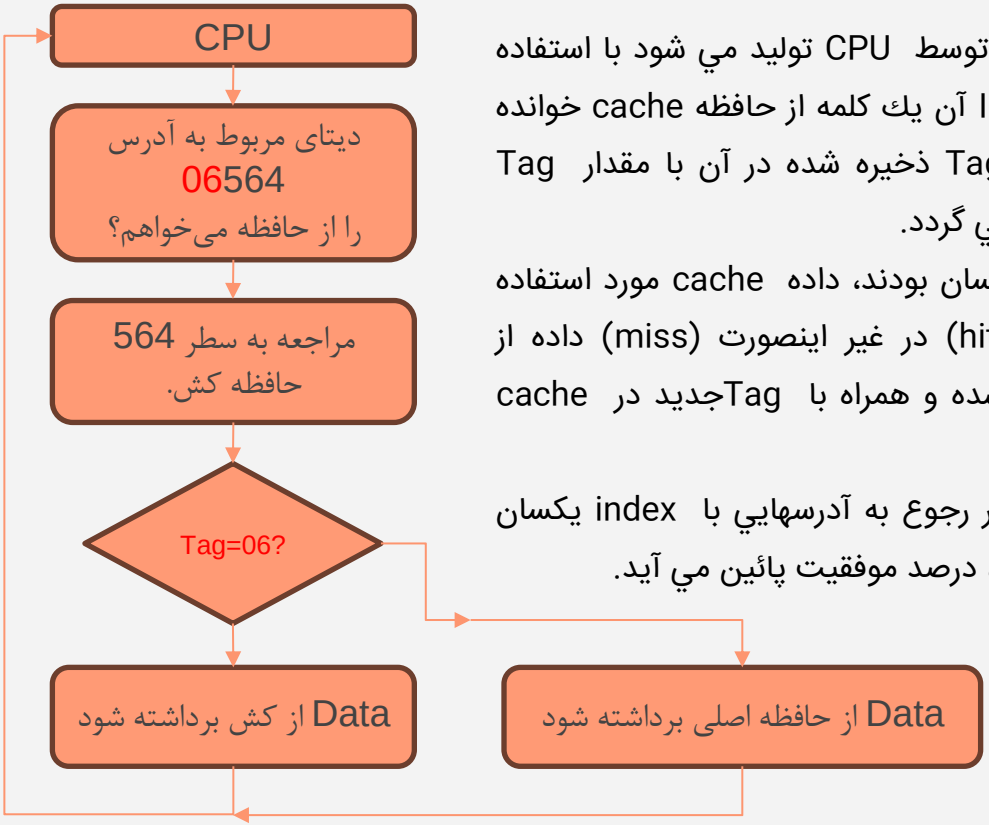
نگاشت مستقیم (Direct Mapping)

- در روش نگاشت مستقیم، اطلاعات و Tag هر دو در حافظه کش ذخیره مي شود.



نگاشت مستقیم (Direct Mapping)

- وقتي كه آدرسي توسط CPU توليد مي شود با استفاده از قسمت Index آن يك كلمه از حافظه cache خوانده شده و مقدار Tag ذخيره شده در آن با مقدار Tag آدرس مقايسه مي گردد.
- اگر دو Tag يكسان بودند، داده cache مورد استفاده قرار مي گيرد (hit) در غير اينصورت (miss) داده از حافظه خوانده شده و همراه با Tagجديد در cache نوشته مي شود.
- در اين روش اگر رجوع به آدرسهايي با index يكسان زياد اتفاق بيافتد، درصد موفقيت پائين مي آيد.



نگاشت مستقيم (Direct Mapping)

- ✓ معمولا اندازه بلوك بزرگتر از يك در نظر گرفته مي شود:

	Index	Tag	Data
Block 0	000	0 1	3 4 5 0
	007	0 1	6 5 7 8
Block 1	010		
	017		
Block 63	770	0 2	
	777	0 2	6 7 1 0

6	6	3
Tag	Block	Word

نگاشت شرکت پذیر (Associate mapping)

- ✓ در این روش که سریعترین راه پیاده سازي يك cache است از يك حافظه associative استفاده مي شود.
- ✓ در این حافظه هم آدرس (به طور کامل) و هم محتواي يك كلمه ذخيره مي شوند.
- ✓ هنگام جستجو براي يك داده، آدرس آن به حافظه associative عرضه شده و در صورتي كه ورودي متناظري در حافظه باشد، داده مربوطه در خروجي ظاهر مي گردد. در غير اينصورت هر دوي آدرس و داده در حافظه associative ذخيره خواهند شد.
- ✓ معمولاً از الگوريتم FIFO براي جايگزيني در cache استفاده مي شود.

نگاشت مجموعه هاي هم پيوند (Set-associative mapping)

- ✓ در این روش حافظه کش با ساختاری مشابه روش نگاشت مستقیم یعنی شامل Tag و Data در هر محل از حافظه cache مي‌تواند پر شود. با این تفاوت که در هر سطر حافظه کش چندین کلمه با آدرس Index یکسان ذخیره می‌شود.
- ✓ تعداد data-tag هاي ذخيره شده در حافظه يك set خوانده مي شود.
- ✓ براي مقايسه tag آدرس توليد شده با مقادير ذخيره شده ازحافظه associative استفاده مي شود.
- ✓ با بزرگ شدن set ها درصد موفقيت cache افزايش مي يابد.

Index	Tag	Data	Tag	Data
000	0 1	3 4 5 0	0 2	5 6 7 0
777	0 2	6 7 1 0	0 0	2 3 4 0

نگاشت مجموعه هاي هم پيوند (Set-associative mapping)

- ✓ وقتي كه يك miss اتفاق مي افتد، در صورتيكه يك set پر باشد لازم مي شود تا يكي از داده هاي آن خالي شده و داده جديد وارد cache شود.

- ✓ روشهاي مختلفي براي جايگزينی داده جديد از حافظه اصلی در حافظه كش:

  - first-in, first-out
  - Random replacement
  - Least recently used (LRU)

نوشتن در حافظه cache

- ✓ يكي از مسايل مهم در ارتباط با حافظه cache نحوه عمل در هنگام نوشتن اطلاعات در حافظه است.

- ✓ هنگام خواندن داده ها وقتي داده در cache وجود داشته باشد، نيازي به رجوع به حافظه اصلي نيست اما وقتي داده را در حافظه مي نويسيم ممكن است به دو طريق عمل شود:

۱. write through

در اين روش درهر بار نوشتن در حافظه داده هم در cache و هم در حافظه اصلي نوشته مي شود.

۲. write back

در اين روش داده فقط در cache نوشته شده و با يك پرچم set مي شود.تا زمانيكه اين داده در cache قرار دارد از اين داده استفاده خواهد شد، اما در صورت انتقال داده از cache مقدار آن در حافظه اصلي نيز update مي گردد.

razeghizade@gmail.com

Razeghizade.pudica.ir

CREADITS: This presentation was created by M.Razeghizadeh
Please keep this slide for attribution