معماری کامپیوتر

سازمان حافظه

فصل دوازدهم کتاب موریس مانو

محمدعلی شفیعیان

http://shafieian-education.ir

بهار ۹۸

سازمان حافظه

سلسله مراتب حافظه

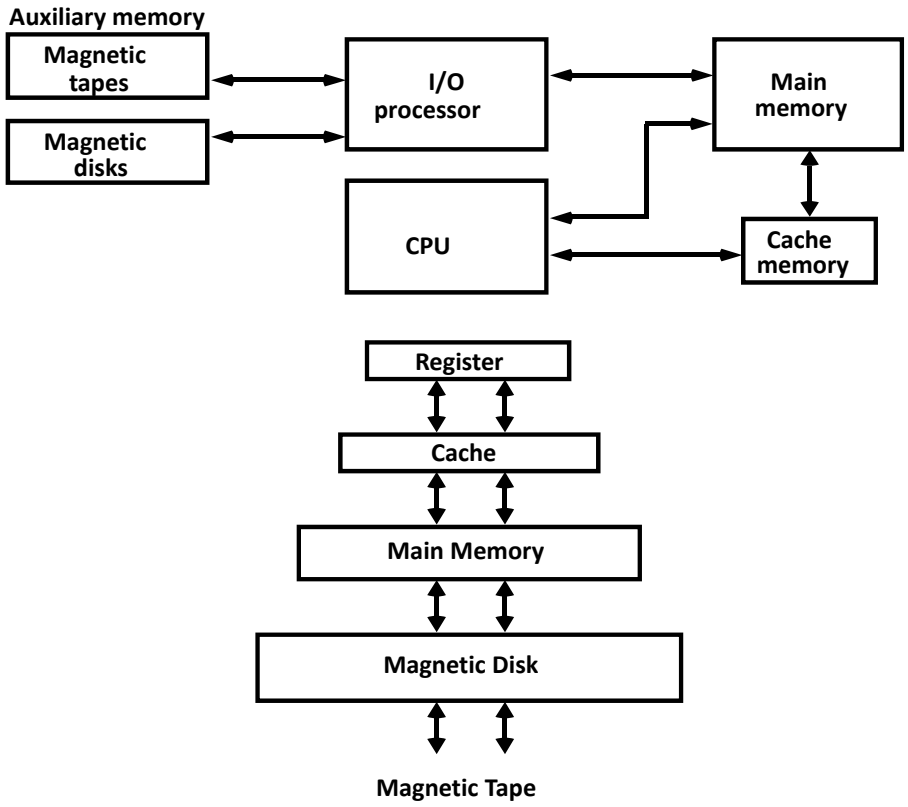
حافظه اصلی

حافظه های کمکی

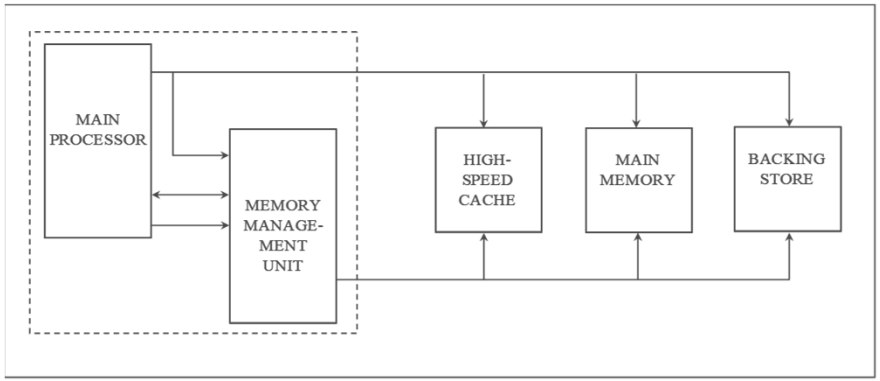
حافظه‌های شرکت‌پذیر یا هم‌پیوند

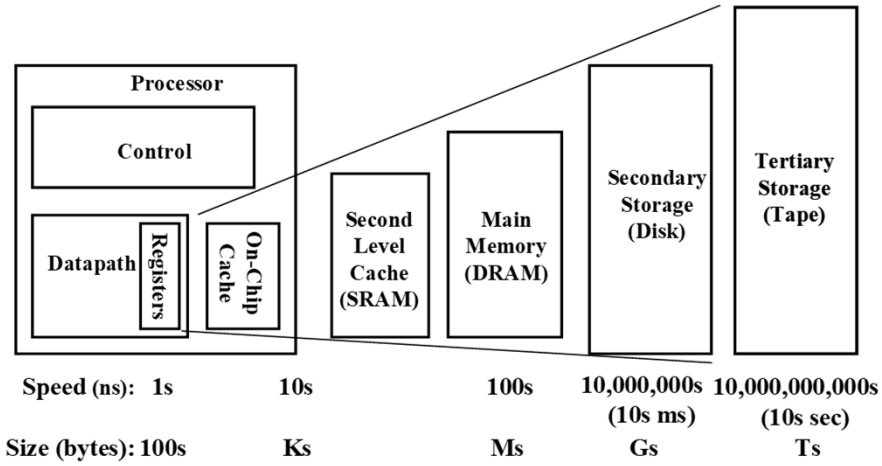
حافظه نهان

سلسله مراتب حافظه

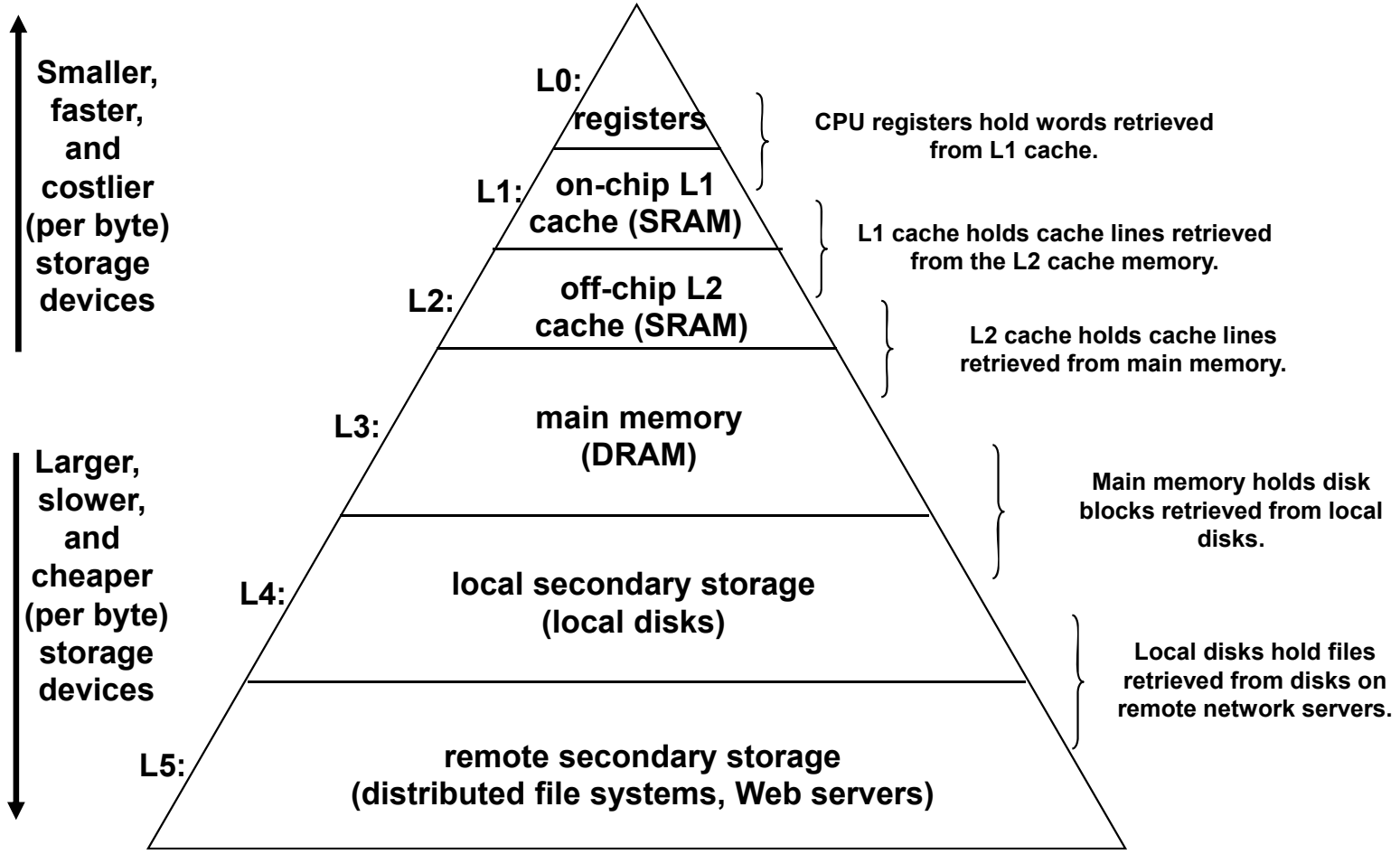


سلسله مراتب حافظه





مثالی از سلسله مراتب حافظه



سلسله مراتب حافظه

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

- **در سلسله مراتب حافظه هرم(که قاعدهٔ آن در پائین است) اگر از پائین به بالا حرکت کنیم:**
  - **سرعت دسترسی به حافظه بیشتر می‌شود.**
  - **هزینه سخت‌افزار افزایش پیدا می‌کند.**
  - **حجم حافظه کاهش پیدا می‌کند.**

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 :**

1. ROM
2. RAM

حافظه ROM

- **حافظه ای است فقط خواندنی که محتوی آن یک بار نوشته شده و پس از نصب در کامپیوتر تغییری در آن داده نمی شود.**

- **معمولا از این حافظه برای ذخیره برنامه هائی نظیر** bootstrap loader **که برای راه اندازی اولیه کامپیوتر مورد نیاز هستند استفاده می شود.**

- **این حافظه انواع مختلفی دارد:**

PROM = Programmable ROM

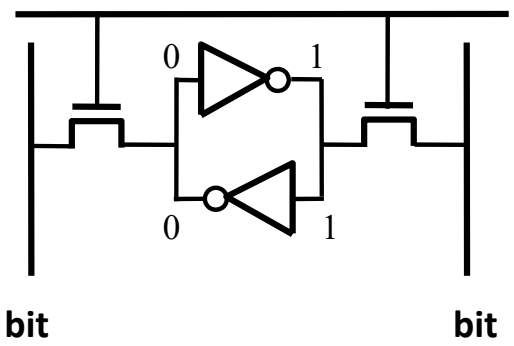
EPROM = Erasable Programmable ROM

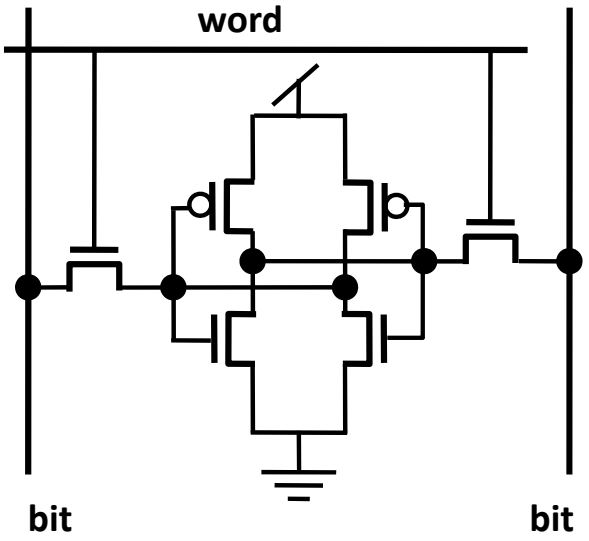
EEPROM = Electrical EPROM

RAM **حافظه**

- **عمده حافظه اصلی کامپیوتر از حافظه** RAM **ساخته می‌شود.**
- **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)

SRAM ساختار حافظه

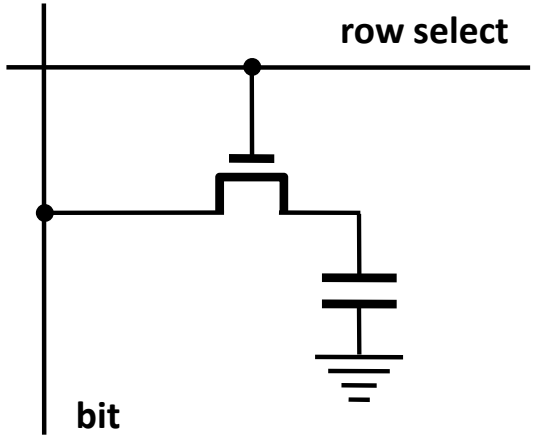




6-Transistor SRAM Cell

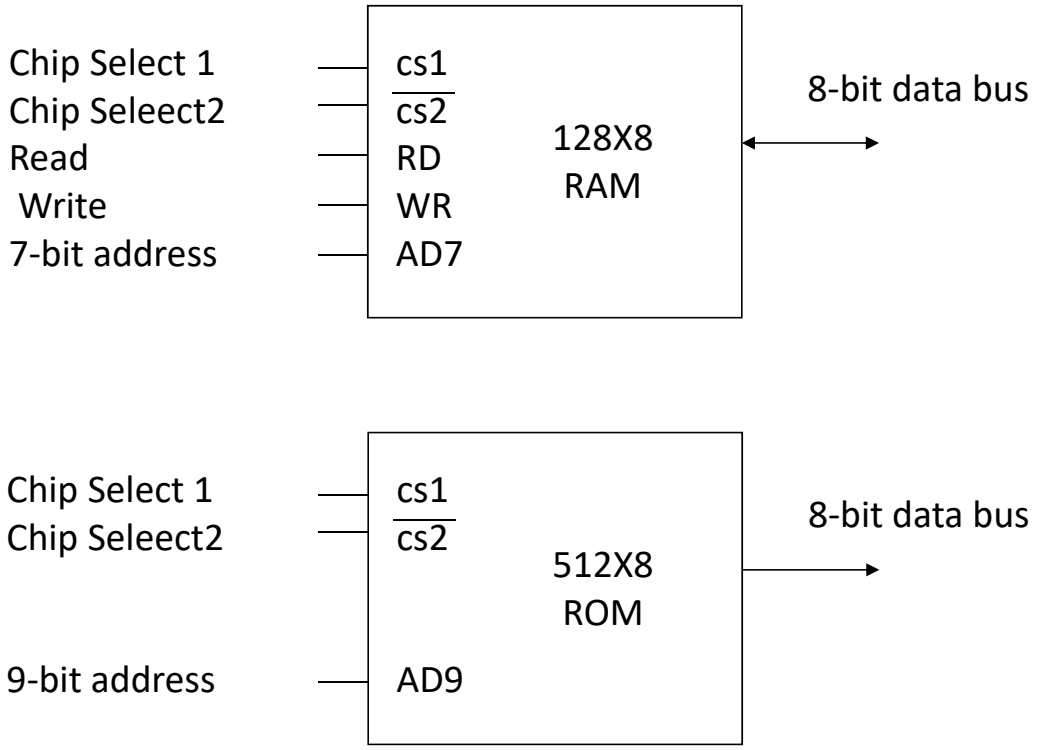
- **Write:**
  1. **Drive bit lines (bit=1, bit=0)**
  2. **Select row**
- **Read:**
  1. **Precharge bit and bit to Vdd or Vdd/2 => make sure equal!**
  2. **Select row**
  3. **Cell pulls one line low**
  4. **Sense amplifier on column detects difference between bit and bit**

DRAM ساختار حافظه



- **Write:**
  - 1. Drive bit line
  - 2. Select row
- **Read:**
  - 1. Precharge bit line to Vdd/2
  - 2. Select row
  - 3. Cell and bit line share charges
    - Very small voltage changes on the bit line
  - 4. Sense (fancy sense amp)
    - Can detect changes of ~10^6 electrons
  - 5. Write: restore the value
- **Refresh**
  - 1. Just do a dummy read to every cell.

بلوک دیاگرام حافظه



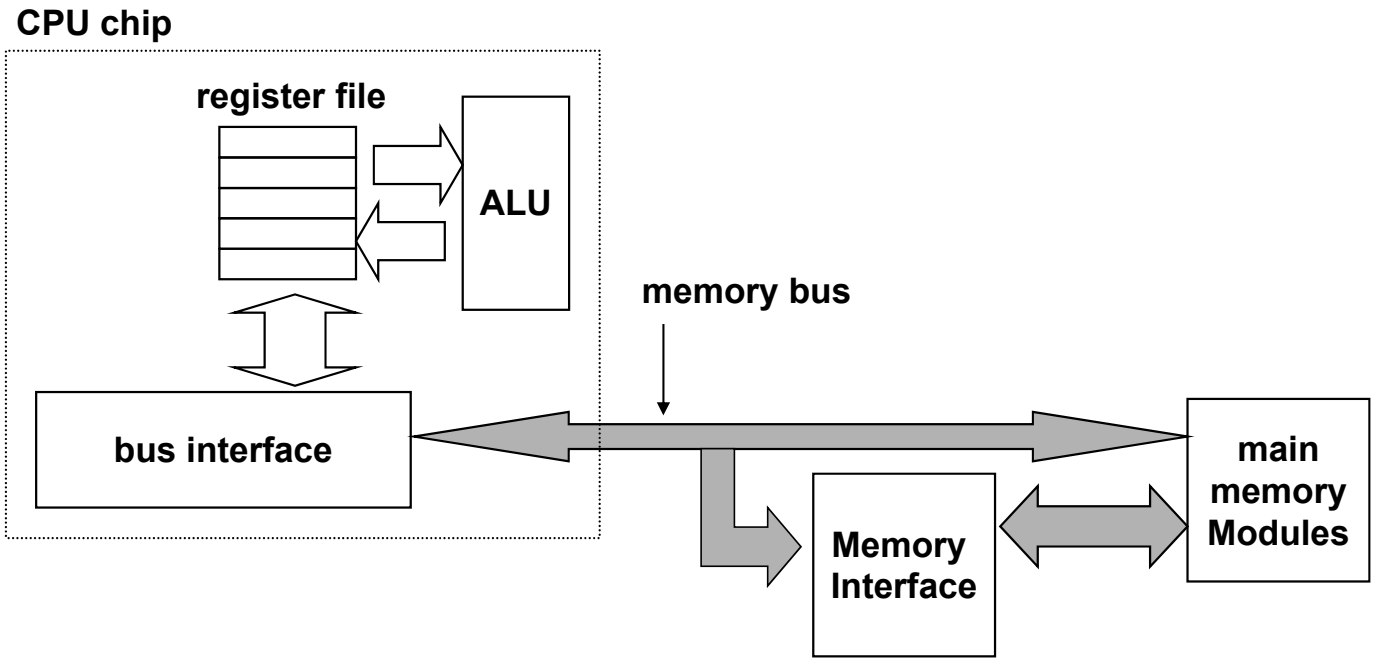
نقشه آدرس حافظه

- در موقع طراحی کامپیوتر مقدار حافظه اختصاص داده شده به هر یک از حافظه های RAM, ROM مشخص می‌گردند.

- مثال :

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	0 x x x x x x x x x

ارتباط حافظه با پردازنده



حافظه اهمیت

- حافظه از چه نظر اهمیت دارد؟

- سرعت دسترسی به حافظه یک گلوگاه اصلی برای افزایش کارائی کامپیوتر است

پیشرفت تکنولوژی حافظه

- در حالیکه در سالیان گذشته حجم حافظه به سرعت افزایش پیدا کرده است، کاهش در زمان دسترسی به محتوی حافظه پیشرفت بسیار کمتری داشته اشت.

1000:1!

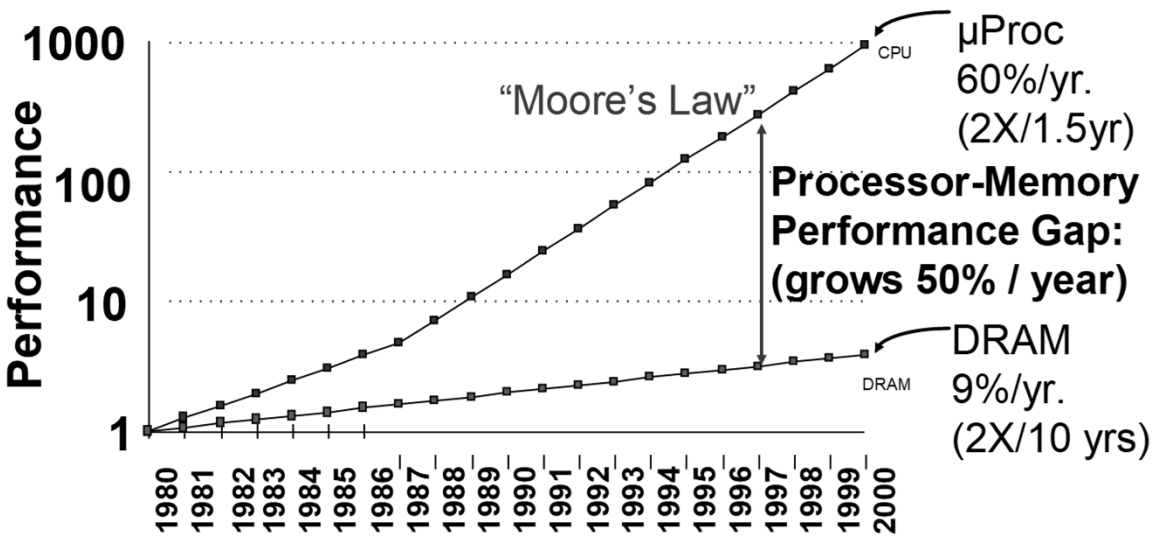
2:1!

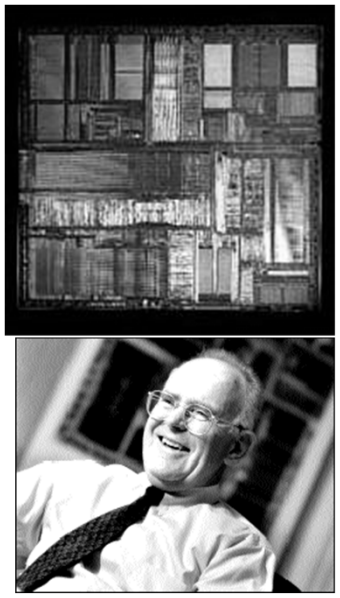
DRAM

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

قانون مور – یک فرصت از دست رفته

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





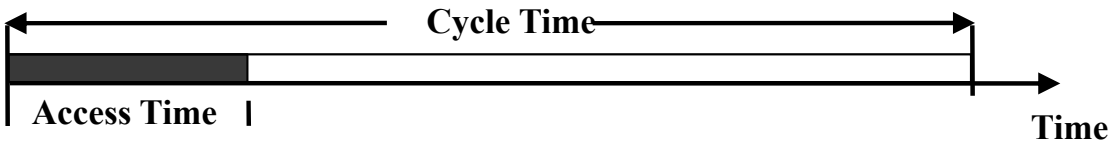
صورت مسئله

- **ویژگیهای حافظه :**
  - **حافظه های بزرگ کند هستند**
  - **حافظه های سریع کوچک هستند**

- **سوال: چگونه می توان حافظه بزرگ سریع و ارزانی داشت؟**
  1. **استفاده از سلسله مراتب**
  2. **دسترسی موازی**

زمان دسترسی به حافظه

- **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 یک حافظه** DRAM **ممکن است** 60ns **باشد با احتساب زمان های لازم برای انتقال آدرس از پردازنده به حافظه، تاخیر باس ها، و ... زمان سیکل آن ممکن است به** 180-250 ns **برسد.**

حافظه شرکت پذیر یا هم پیوند
Associative Memory (AM)

- **از این حافظه برای تسریع عمل جستجو در بین داده های ذخیره شده استفاده می شود.**

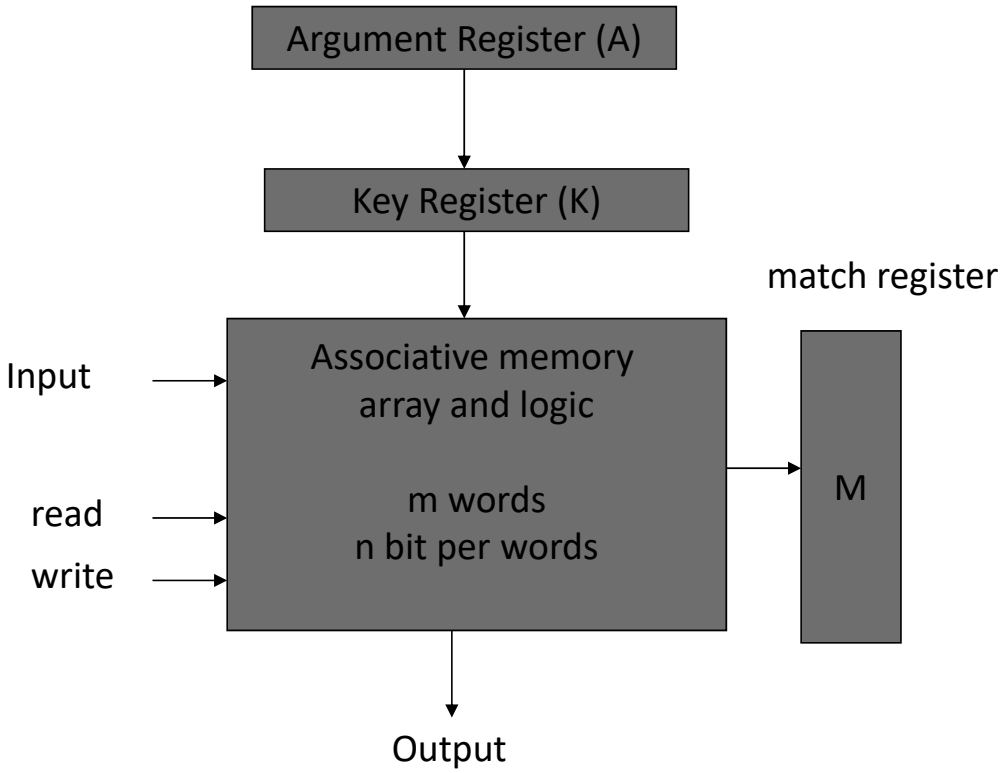
- **برای کاهش زمان جستجو به جای استفاده از آدرس، از محتوی خود داده استفاده می شود. این روش را گاهی** Content Addressable Memory (CAM) **هم می نامند.**

- **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.**

مربوطه افزار سخت



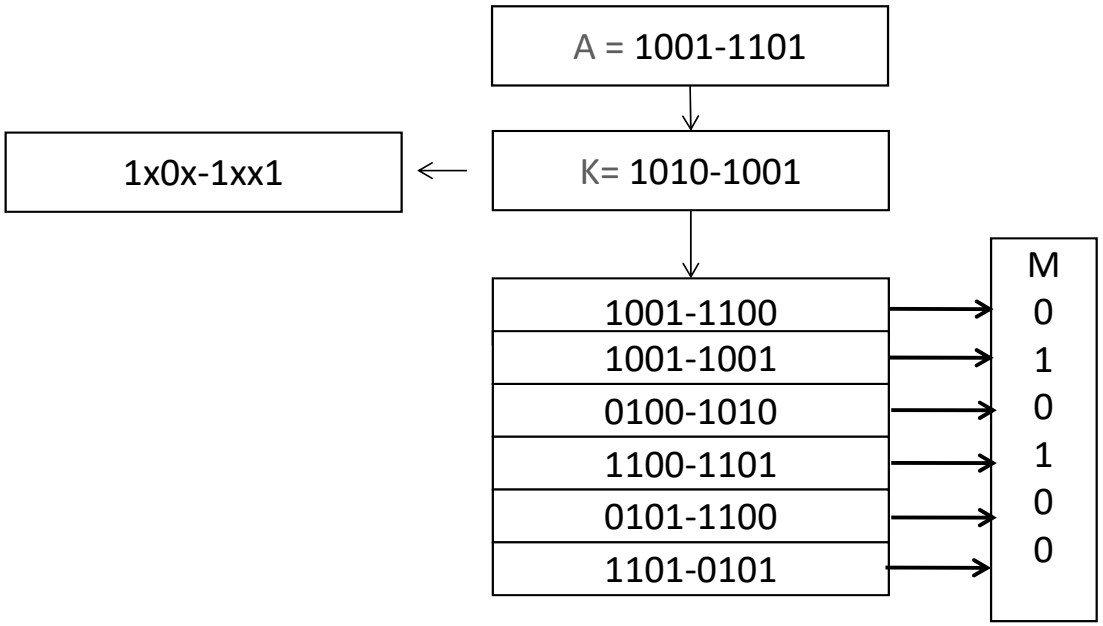
عملکرد حافظه شرکت پذیر

- هر کلمه در حافظه به طور موازی با محتویات ثبات A (Argument Register) مقایسه می‌شود، البته ثبات کلید می تواند محدوده جستجو را کاهش دهد.

- اگر در حافظه، word[i]=A آنگاه M[i]=1.

- تمام کلماتی که بیت متناظر آنها در M یک شده باشد، از حافظه خوانده می شوند.

مثال

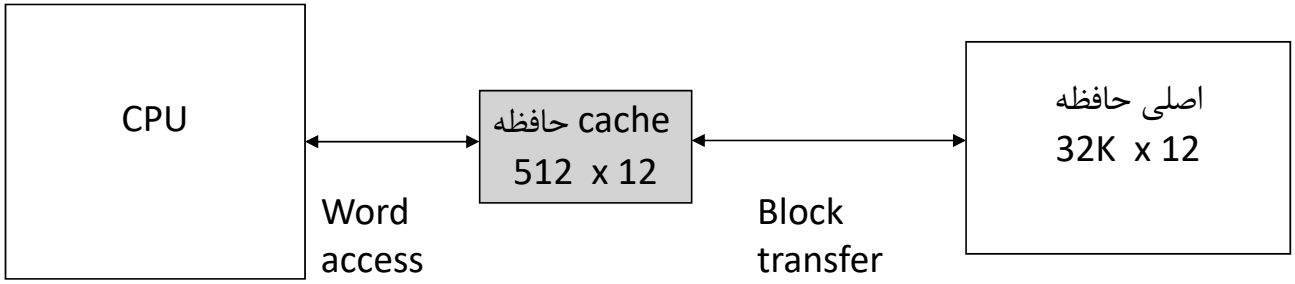

A = 1001-1101
K= 1010-1001
1x0x-1xx1
1001-1100
1001-1001
0100-1010
1100-1101
0101-1100
1101-0101
M
0
1
0
1
0
0

حافظه نهان (cache)

- اگر قسمت فعال برنامه و داده ها را در حافظه سریع و کوچکی قرار دهیم، می توانیم با کم کردن میانگین زمان دسترسی به حافظه زمان اجرای برنامه را کاهش دهیم.
- این حافظه سریع و کوچک را حافظه cache می نامند.
- معمولا حافظه cache زمان دسترسی بسیار کمتری نسبت به حافظه اصلی دارد (۵ تا ۱۰ برابر کمتر)
- دریک کامپیوتر ممکن است عمل cache تا چند سطح تکرار گردد.

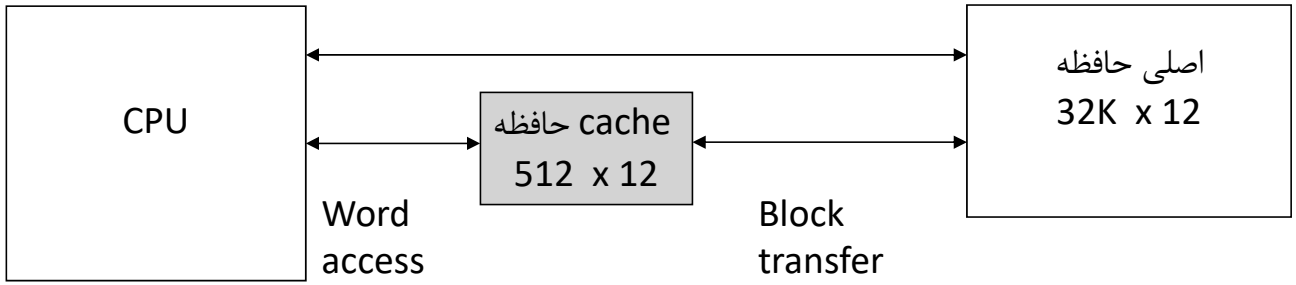
نحوه عمل cache

- **وقتی که CPU نیاز به دسترسی به حافظه دارد ابتدا حافظه cache را جستجو می نماید؛ اگر داده در این حافظه سریع موجود بود (hit)، از آن استفاده می شود در غیر اینصورت (miss)، با رجوع به حافظه بلوکی از داده که شامل داده مورد نیاز CPU می‌شود از حافظه اصلی به حافظه cache منتقل می گردد.**



نحوه عمل cache

- **ممکن است CPU منتظر نوشته شدن داده و اطراف آن (یک بلوک) در حافظه نهان نباشد و داده را مستقیما از حافظه اصلی دریافت کند.**

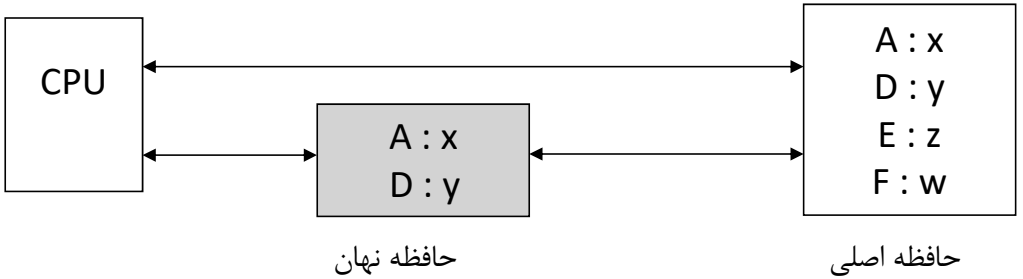


درصد موفقیت

- کارایی حافظه cache با ضریبی با نام درصد موفقیت (نرخ دسترسی) hit ratio اندازه گیری می شود.
- این درصد عبارت است از نسبت تعداد دفعاتی که داده در cache یافت شده به تعداد کل دفعات رجوع به حافظه.

$$h = \frac{\text{تعداد hit ها}}{\text{تعداد کل مراجعات}}$$

- هر چه مقدار این درصد بیشتر باشد سرعت دسترسی به حافظه به سرعت cache نزدیکتر می‌شود.



زمان متوسط دسترسی

الف : با فرض اینکه حافظه اصلی می تواند مستقیما به CPU داده بدهد :

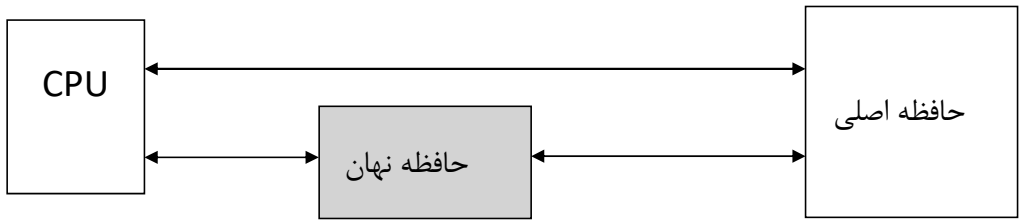
t_{eff} : **زمان دستیابی متوسط (موثر)** cache

t_{cache} : **زمان دستیابی** cache

t_{main} : **زمان دستیابی حافظه اصلی**

h: **نسبت برخورد(موفقیت)**

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



مثال

یک کامپیوتر با زمان دسترسی 100ns برای حافظه cache و1000ns برای حافظه اصلی در صورت داشتن درصد موفقیت 0.9 زمان دسترسی برابر با 190 ns خواهد داشت (فرض اینکه حافظه اصلی می تواند مستقیما به CPU داده بدهد).

$$t_{cache} = 100\ ns$$

$$t_{main} = 1000\ ns$$

$$h = 0.9$$

$$\Rightarrow \quad t_{avg} = t_{eff} = (0.9 \times 100) + (1 - 0.9) \times 1000 = 190\ ns$$

زمان متوسط دسترسی

ب: با فرض اینکه حافظه اصلی مستقیما به CPU داده ندهد :

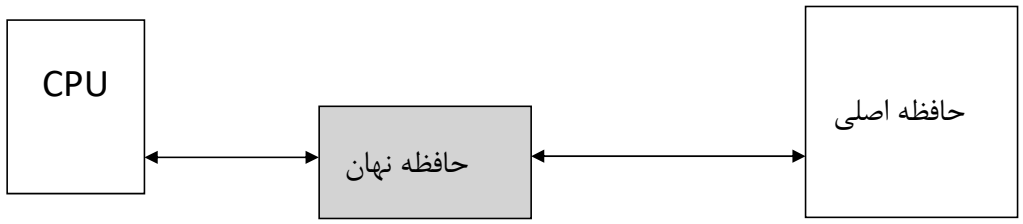
t_{eff} : **زمان دستیابی متوسط (موثر)** cache

t_{cache} : **زمان دستیابی** cache

t_{main} : **زمان دستیابی حافظه اصلی**

h: **نسبت برخورد(موفقیت)**

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



مثال

یک کامپیوتر با زمان دسترسی 100ns برای حافظه cache و1000ns برای حافظه اصلی در صورت داشتن درصد موفقیت 0.9 زمان دسترسی برابر با 200 ns خواهد داشت (فرض اینکه حافظه اصلی مستقیما به CPU داده ندهد).

$$t_{cache} = 100\ ns$$

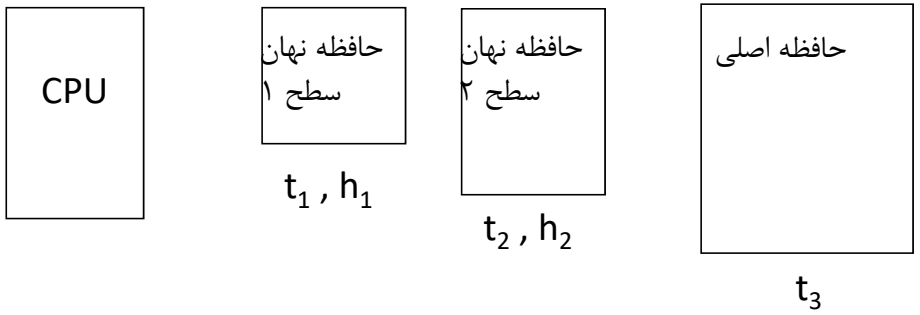
$$t_{main} = 1000\ ns$$

$$h = 0.9$$

$$\Rightarrow \quad t_{avg} = t_{eff} = 100 + (1 - 0.9) \times 1000 = 200\ ns$$

زمان متوسط دسترسی (تعمیم)

- **فرمول های پیش را می توان به چندین سطح حافظه نیز گسترش داد :**



- **الف) اگر CPU مستقیما به حافظه متصل نباشد :**

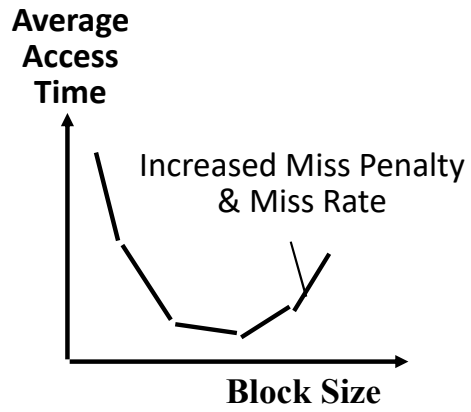
$$t_{avg} = t_{eff} = t_1 + (1 - h_1)t_2 + (1 - h_1)\,(1 - h_2)t_3$$

- **ب) اگر CPU مستقیما به حافظه متصل باشد :**

$$t_{avg} = t_{eff} = h_1t_1 + (1 - h_1)h_2t_2 + (1 - h_1)\,(1 - h_2)t_3$$

تاثیر اندازه بلوک‌های cache

- در حالت کلی با بزرگ شدن اندازه بلوک های cache میزان hit rate افزایش می یابد. اما به همین ترتیب میزانmiss penalty یعنی زمانی که برای انتقال اطلاعات به حافظه نهان لازم است نیز افزایش خواهد یافت.



Average Access Time = Hit Time x Hit Ratio + Miss Penalty x (1 – Hit Ratio)

نگاشت

- **عمل انتقال داده از حافظه اصلی به حافظه** cache **را نگاشت می نامند .این کار به سه طریق انجام می شود :**

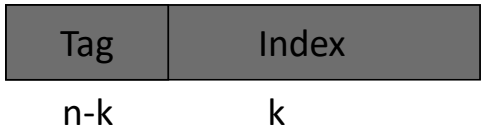
  - **نگاشت مستقیم** (Direct Mapping)
  - **نگاشت شرکت پذیر** (Associate Mapping)
  - **نگاشت مجموعه های هم پیوند** (Set-associative mapping)

نگاشت مستقیم (Direct Mapping)

- در این روش برای پیاده سازی cache از یک حافظه RAM سریع استفاده می شود:

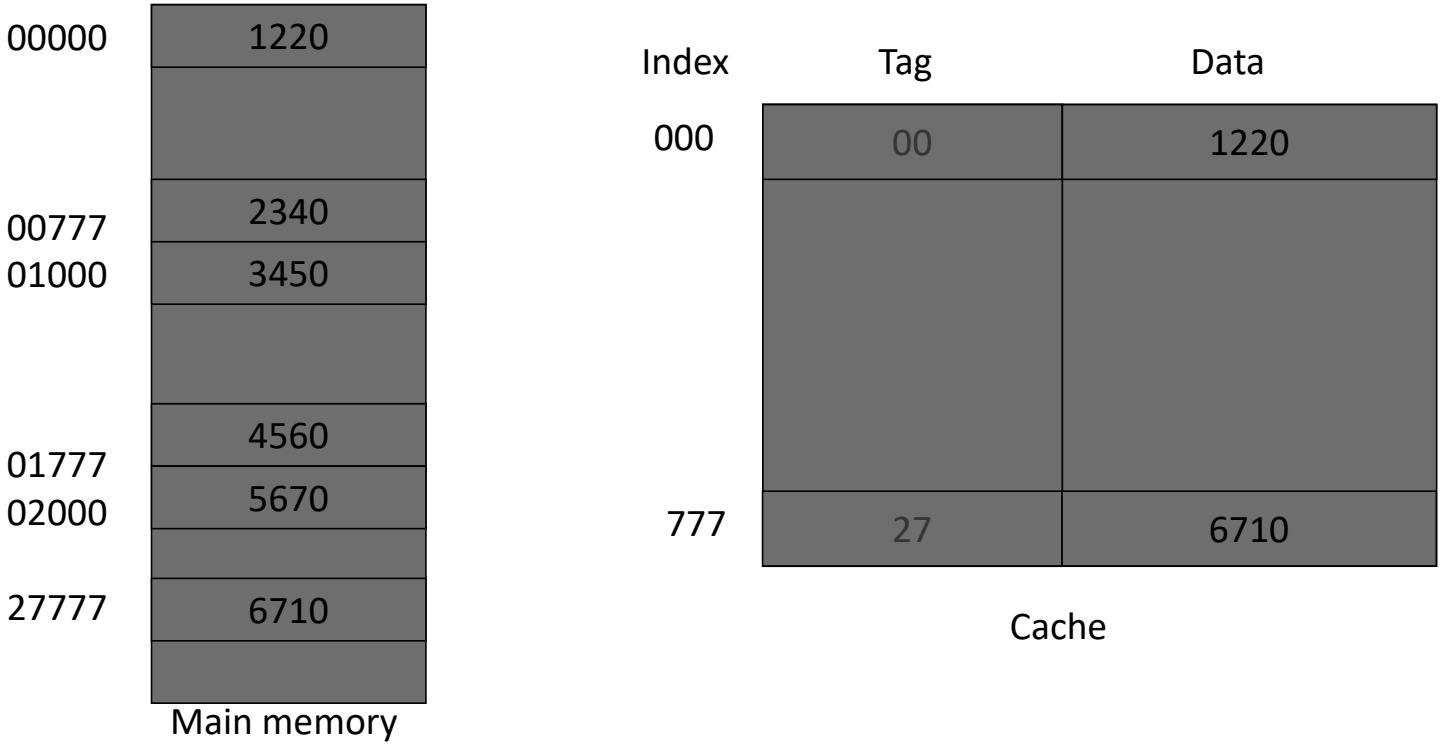
- اگر حافظه اصلی دارای 2^n کلمه و حافظه cache دارای 2^k کلمه باشند در نتیجه به ترتیب به n و k بیت آدرس نیاز خواهند داشت.

- آدرس n بیتی حافظه اصلی به صورت زیر تقسیم می شود:



نگاشت مستقیم (Direct Mapping)

- در حافظه cache علاوه بر داده اطلاعات مربوط به Tag هم ذخیره می شود.



نگاشت مستقیم (Direct Mapping)

- وقتی که آدرسی توسط CPU تولید می شود با استفاده از قسمت Index آن یک کلمه از حافظه cache خوانده شده و مقدار Tag ذخیره شده در آن با مقدار Tag آدرس مقایسه می گردد.

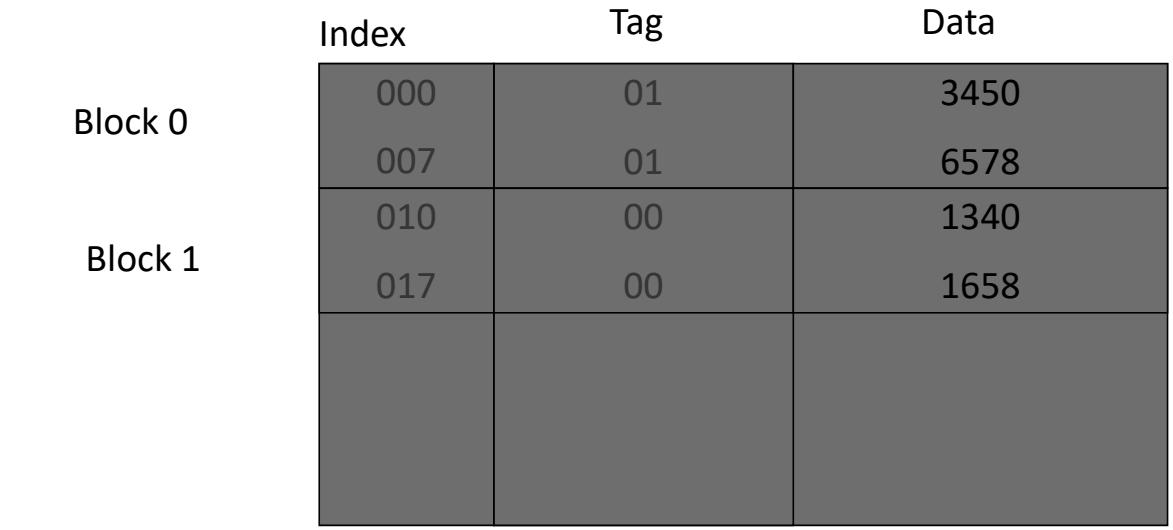
- اگر دو Tag یکسان بودند، داده cache مورد استفاده قرار می‌گیرد(hit) .

- در غیر اینصورت (miss) داده از حافظه خوانده شده و همراه با Tag جدید در cache نوشته می شود.

- در این روش اگر رجوع به آدرسهائی با index یکسان زیاد اتفاق بیافتد، درصد موفقیت پائین می آید.

نگاشت مستقیم با انداز بلوک بزرگتر

- معمولا اندازه بلوک بزرگتر از یک در نظر گرفته می شود:



نگاشت شرکت پذیر
(Associate mapping)

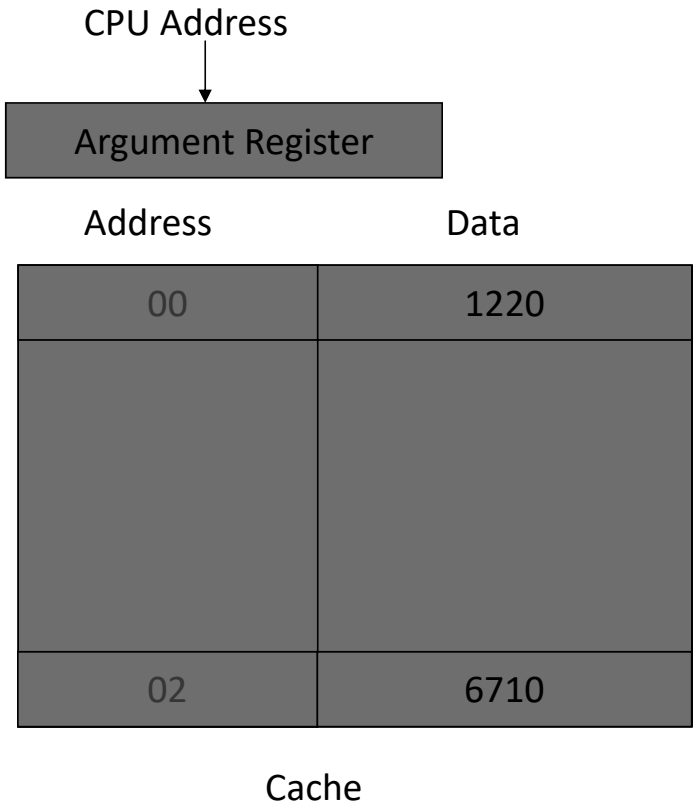
- در این روش که سریعترین راه پیاده سازی یک cache است از یک حافظه associative استفاده می شود.

- در این حافظه هم آدرس و هم محتوای یک کلمه ذخیره می شوند. در نتیجه cache می تواند محتوای هر محل از حافظه را ذخیره نماید.

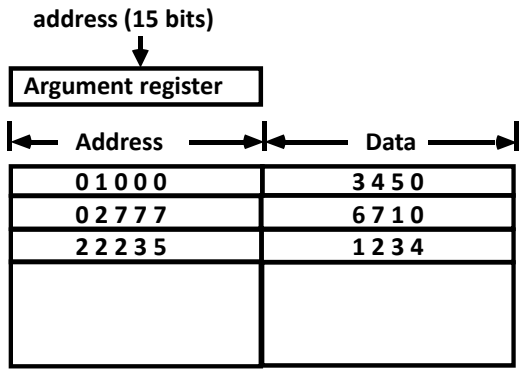
- هنگام جستجو برای یک داده آدرس آن به حافظه associative عرضه می شود. در صورتی که ورودی متناظری در حافظه باشد، داده مربوطه در خروجی ظاهر می گردد. در غیر اینصورت هر دوی آدرس و داده در حافظه associative ذخیره خواهند شد.

- معمولا از الگوریتم FIFOبرای جایگزینی در cache استفاده می شود.

نگاشت شرکت پذیر
(Associate mapping)



نگاشت شرکت پذیر
(Associate mapping)

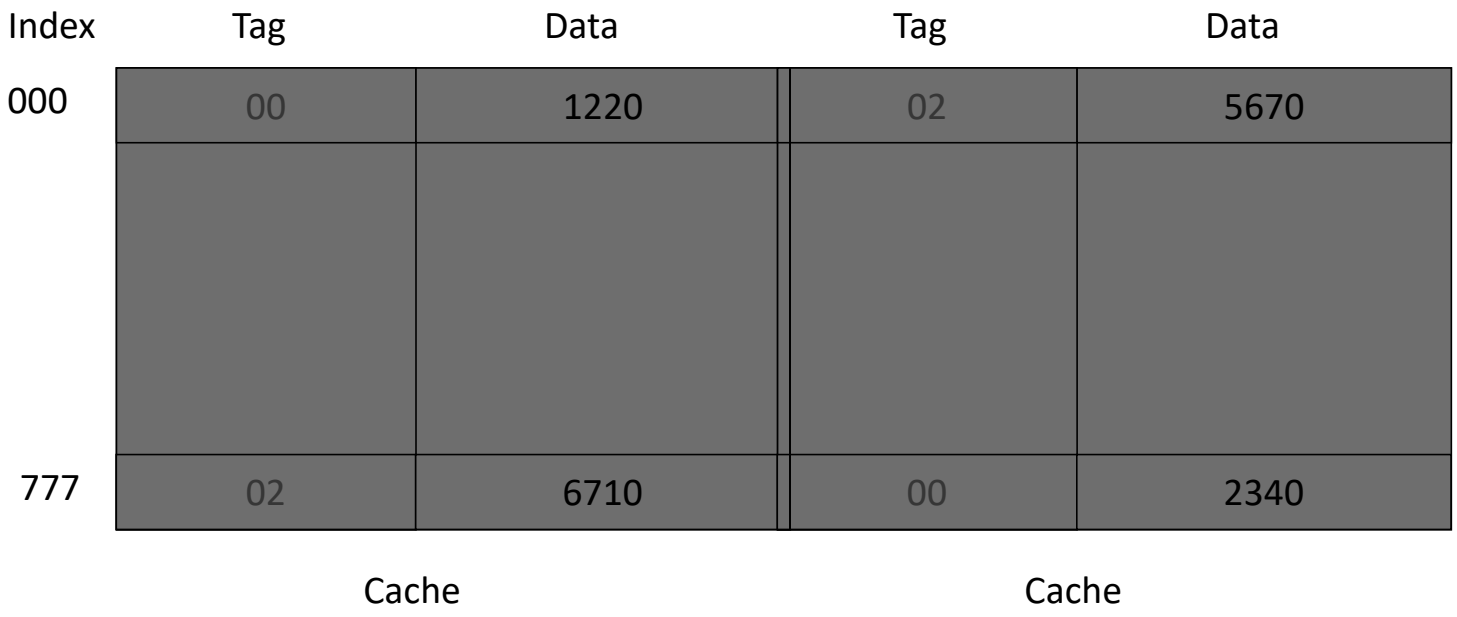


حافظه شرکت پذیر

نگاشت مجموعه های هم پیوند (Set-associative mapping)

- در این روش هر محل از حافظه cache میتواند چندین کلمه با آدرس Index یکسان را ذخیره نماید.
- تعداد data-tag های ذخیره شده در حافظه یک set خوانده می شود.
- برای مقایسه tag آدرس تولید شده با مقادیر ذخیره شده ازحافظه associative استفاده می شود.
- با بزرگ شدن set ها درصد موفقیت cache افزایش می یابد.

نگاشت مجموعه های هم پیوند
(Set-associative mapping)



جایگزینی های الگوریتم

- وقتی که یک miss اتفاق می افتد، در صورتیکه یک set پر باشد لازم می شود تا یکی از داده های آن خالی شده و داده جدید وارد cache شود.

- روشهای مختلفی برای این کار وجود دارد:
  1. first-in, first-out
  2. Random replacement
  3. Least recently used (LRU)

نوشتن در حافظه cache

- یکی از مسایل مهم در ارتباط با حافظه cache نحوه عمل در هنگام نوشتن اطلاعات در حافظه است.

- هنگام خواندن داده ها وقتی داده در cache وجود داشته باشد، نیازی به رجوع به حافظه اصلی نیست اما وقتی داده را در حافظه می نویسیم ممکن است به دو طریق عمل شود:

1. ***write through*** : در این روش درهر بار نوشتن در حافظه داده هم در cache و هم در حافظه اصلی نوشته می شود.
2. ***write back*** :در این روش داده فقط در cache نوشته شده و با یک پرچم set می شود.تا زمانیکه این داده در cache قرار دارد از این داده استفاده خواهد شد، اما در صورت انتقال داده از cache مقدار آن در حافظه اصلی نیز update می گردد.

چه موقع write through بهتر است؟

- زمانی که از cache دو سطحی استفاده می شود
  - سازگاری آسانتر است.
  - به وسیله دومین cacheاز ترافیک حافظه جلوگیری می شود.
  - قابلیت اطمینان (write-throughبهتر است) برای اینکه حافظه اصلی قابلیت شناسایی خطا را دارد.
  - سازگاری cache (write-through بهتر است).

حافظه cache در 80486

- در داخل این پردازنده یک کاشه 8k بایتی در نظر گرفته شده است و از بلوک های 16 بایتی استفاده می شود.
- برای نوشتن در حافظه نهان از روش write through استفاده می‌شود.
- درصد موفقیت بالاتر از 90 در صد است.
- می توان از یک حافظه نهان خارجی دیگر نیز استفاده نمود.

سلسله مراتب حافظه در پردازنده های جدید

Intel Pentium 4, 2.2 GHz Processor.

Component	Access Speed (Time for data to be returned)	Size of Component
Registers	1 cycle = 0.5 nanoseconds	32 registers
L1 Cache	3 cycles = 1.5 nanoseconds	Separate Data and Instruction Caches: 8 Kbytes each
L2 Cache	20 cycles = 10 nanoseconds	256 Kbytes, 8-way set associative
L3 Cache	30 cycles = 15 nanoseconds	512 Kbytes, 8-way set associative
Memory	400 cycles = 200 nanoseconds	16 Gigabytes

حافظه مجازی

- برنامه ها به اندازه حافظه اصلی محدود می شوند.

- فقط باید در این محدوده آدرس دهی کنند و با توجه به ظرفیت پایین حافظه اصلی، این برای بعضی برنامه ها مشکل ساز است.

- برای این منظور از حافظه مجازی استفاده می شود، ظرفیت این حافظه در واقع مجموع حافظه اصلی و حافظه کمکی است.

- حافظهٔ مجازی به کاربر این دید را می‌دهد که سیستم دارای یک حافظهٔ بسیار بزرگ است،هرچند واقعاً این طور نباشد و حجم حافظهٔ اصلی بسیار کوچک باشد.

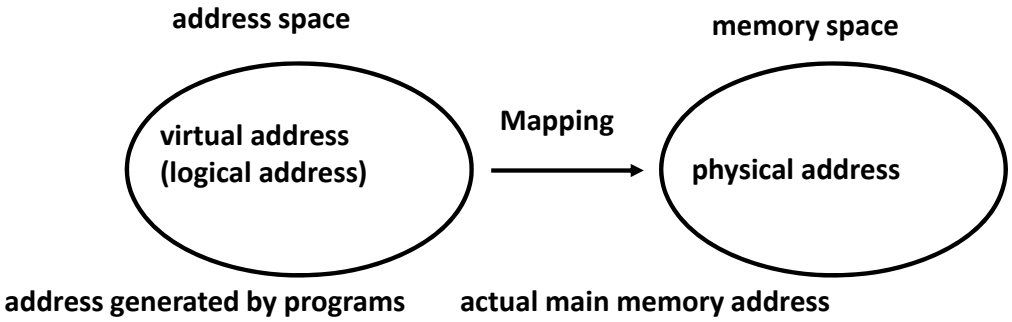
حافظه مجازی

- برنامه ها، با توجه به ظرفیت حافظه مجازی، نوشته می شوند.

- در آدرس دهی ها، به حافظه مجازی آدرس داده می شود که ظرفیت بیشتری نسبت به حافظه اصلی دارد استفاده می شود. این فضای آدرس دهی است. این آدرسها مجازی هستند و آدرس حافظه اصلی را مشخص نمی کنند.

- اما برای اعمال به حافظه اصلی باید این آدرس به یک آدرس واقعی و فیزیکی تبدیل شود. آدرسهای قابل اعمال به حافظه اصلی، فضای حافظه هستند.

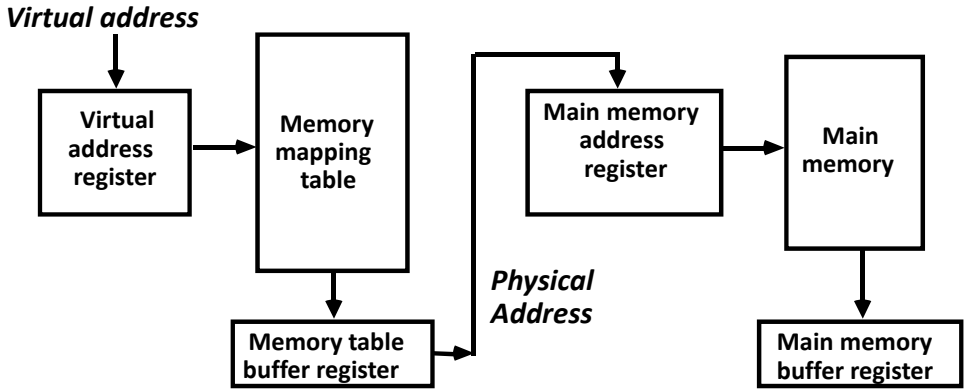
حافظه مجازی


address space
memory space
virtual address
(logical address)
Mapping
physical address
address generated by programs
actual main memory address

نگاشت آدرس در حافظه مجازی

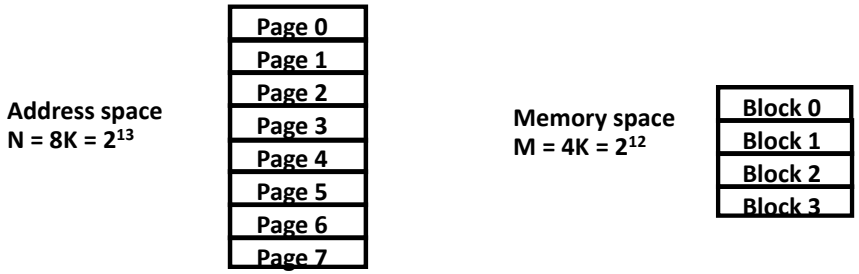
- نگاشت آدرس: جدول نگاشت حافظه برای نگاشت آدرس های مجازی به آدرس های فیزیکی.

- آدرس مجازی نگاشت می شود به یک آدرس واقعی(فیزیکی) و با توجه به آن به حافظه اصلی دسترسی می شود.



نگاشت آدرس در حافظه مجازی

- فضای آدرس و فضای حافظه هر کدام به قسمت های ثابت و مساوی تقسیم می‌شوند.
- فضای آدرس به چندین صفحه تقسیم می شود.
- فضای حافظه به چندین قاب صفحه تقسیم می شود.
- صفحه ها و قاب ها دارای یک اندازه برابر هستند.
- در برنامه ها در هنگام آدرس دهی، ابتدا شماره صفحه را مشخص می کنند و سپس مکان داده در آن صفحه.
- همانطور که در شکل زیر مشاهده می شود، فضای آدرس بیشتر از فضای حافظه است.

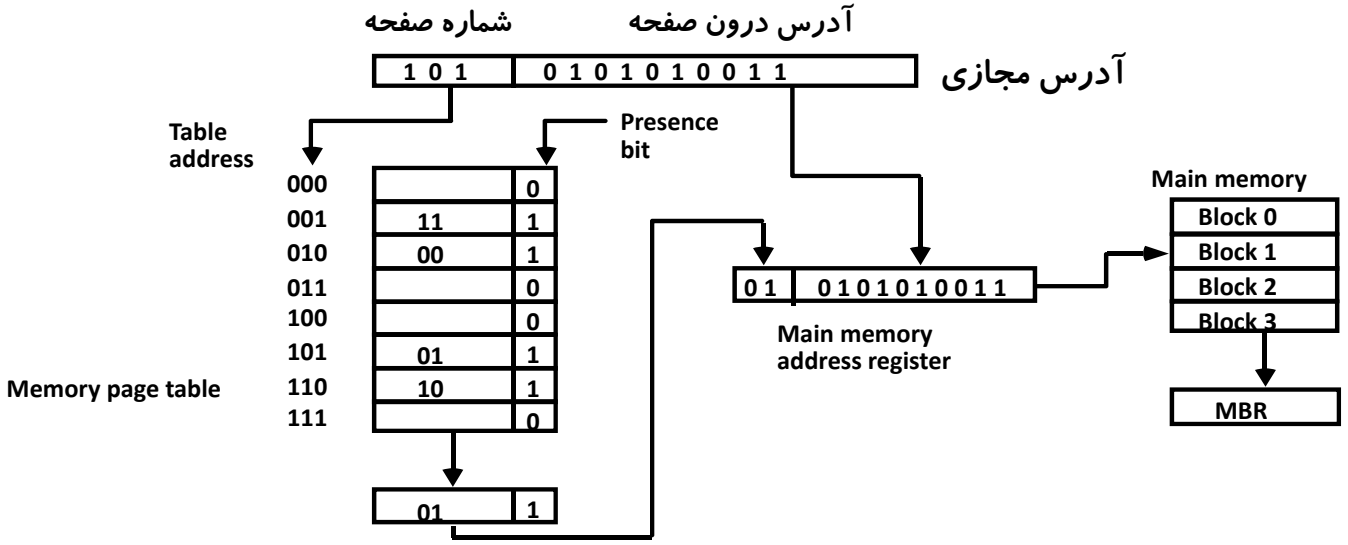


نگاشت آدرس در حافظه مجازی

- باید سخت افزاری وجود داشته باشد، تا هر آدرس فضای مجازی را به فضای فیزیکی تبدیل کند.
- آدرس مجازی شامل دو بخش است، شماره صفحه و آدرس درون صفحه.
- ابتدا شماره صفحه را به یک قاب از حافظه اصلی نگاشت می کنند و سپس در آن قاب آدرس موردنظر را جستجو می کنند.
- می دانیم که ممکن است صفحه موردنظر در حافظه کمکی باشد و در حافظه اصلی موجود نباشد. ابتدا باید بررسی نماییم که صفحه موردنظر در حافظه اصلی قرار دارد یا نه.
- حافظه اصلی محدود است و تنها برای چند صفحه، فضا دارد.
- باید مشخص باشد که چه صفحاتی در حافظه اصلی هستند و در صورت وجود کجای حافظه اصلی هستند.

نگاشت آدرس در حافظه مجازی

- در صورت وجود:
  - با توجه به شماره صفحه، تعیین باید کنیم، که این صفحه در چه قابی از حافظه اصلی است.
  - سپس با آدرس درون صفحه، به مکان خاصی از قاب مربوطه دسترسی می کنیم.
  - پس آدرس فیزیکی، با نگاشت شماره صفحه به شماره قاب موردنظر آن بدست می آید.
  - برای این نگاشت از جدول صفحه استفاده می کنیم(اینجا بیت حضور هم داریم)



__ Robert Fulghum

برای کسب اطلاعات بیشتر در مورد این درس می‌توانید به وب سایت آموزشی در لینک زیر مراجعه نمایید

http://shafieian-education.ir