# Arta - Lec 5

### **(میان ترم)سوال اول (اسلاید 5) الف -** سیاست نگاشت مستقیم (direct mapped) را توضیح دهید. **ب -** نقطه ضعف اصلی این سیاست جایابی چیست ؟

ج - پیشنهاد شما برای حل این نقطه ضعف چیست ؟ (امتیازی)

**پاسخ  
الف -** در این سیاست یک نگاشت یکتا از main memory به حافظه نهان (cache) وجود دارد. یعنی اطلاعات داخل هر خانه حافظه اصلی، پس از انتقال به حافظه نهان، تنها در یک خانه مشخص از حافظه نهان می تواند قرار گیرد.

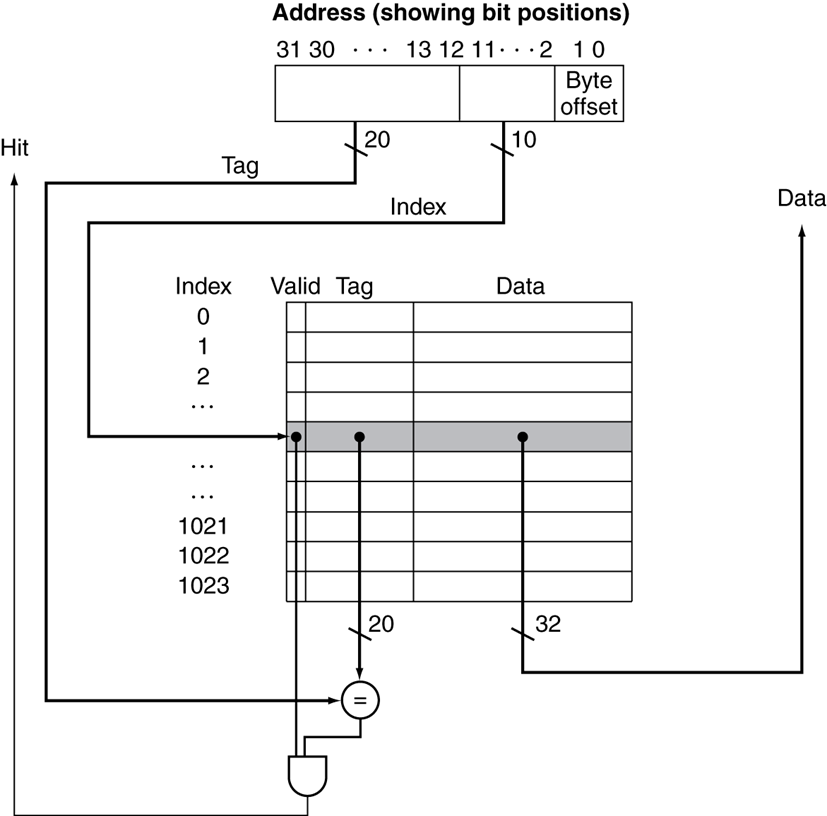
**ب -** در این سیاست با توجه به اینکه هر خانه حافظه اصلی دقیقا به یکی از خانه های حافظه نهان نگاشت می شود، در صورتی که در آن خانه حافظه نهان از پیش مقدار دیگری قرار داشته باشد، حتی با وجود خالی بودن خانه های دیگر، داده جدید جایگزین داده قبلی در خانه مورد نظر می شود.

ج - به جای اینکه برای هر آدرس فقط یک جای مشخص در حافظه نهان موجود باشد، چند جا داشته باشد و هر وقت تمام آن جاها پر شد با یک سیاست خاص تصمیم گرفته شود که کدام یک از آنها جای خود را به بلوک جدید دهد. در واقع به این روش associative گفته می شود و از سیاست های جایگزینی گوناگونی همچون LRU، FIFO و غیره استفاده می شود که جلوتر با آنها آشنا خواهید شد.

### **(میانترم یا کوییز) سوال دوم (اسلاید 5)** فرایند بازیابی یک بلوک داده از طریق کش در سیاست نگاشت مستقیم را توضیح دهید.

**پاسخ**

مطابق شکل زیر، ابتدا از بیت های آدرس دهی، بیت های index که ردیف های حافظه نهان را آدرس دهی می کنند، خانه مورد نظر در cache را تعیین می کنند. قسمت tag آدرس با tag ردیف تعیین شده در حافظه نهان مقایسه می شود و نتیجه با بیت valid ردیف حافظه and می شود و در صورتی که نتیجه برابر با یک باشد hit رخ می دهد و داده از حافظه نهان خوانده می شود و در غیر این صورت miss رخ داده و داده از حافظه اصلی گرفته می شود و در حافظه نهان نیز نوشته می شود.



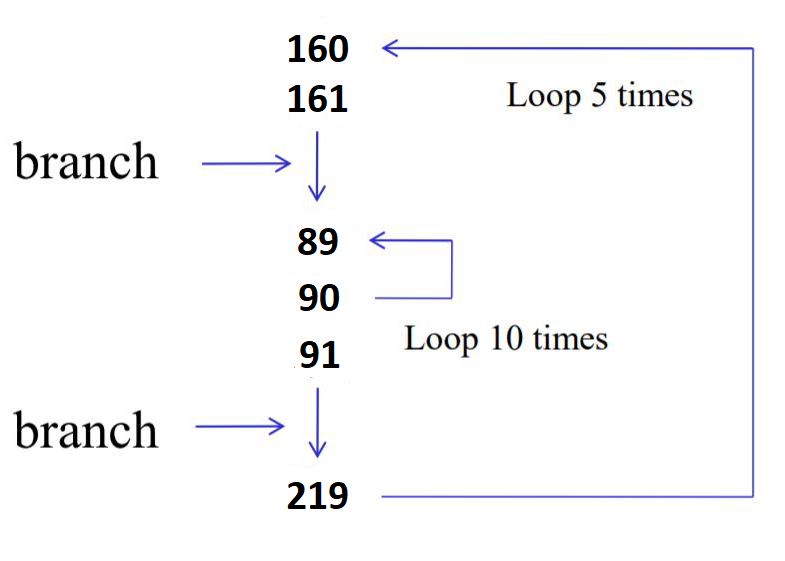
## **(تمرین و یه عددای جدید برا میانترم) سوال سوم (اسلاید 5)** گنجایش یک حافظه اصلی 512 کیلوبایت است و اندازه هر بلوک آن دو word می باشد. در صورتی که گنجایش حافظه نهان 64 بایت بوده و سیاست جایدهی به صورت نگاشت مستقیم باشد قالب آدرس دهی خانه های حافظه را مشخص کنید. (هر word یک بایت است)

**پاسخ**ابتدا تعداد بایت های کل حافظه را حساب می کنیم تا تعداد کل بیت های آدرس دهی مشخص شوند.

در نتیجه تعداد کل بیت های آدرس دهی 19 می باشد. با توجه به اینکه اندازه هر بلوک 16 بیت (2 بایت یا دو کلمه در این سوال) است یک بیت () برای بخش offset نیاز داریم. همین طور با توجه به اینکه هر بلوک 2 بایت است و گنجایش حافظه نهان نیز 64 بایت است، در حافظه نهان 64/2 = 32 بلوک جا می گیرد در نتیجه برای آدرس دهی حافظه نهان نیز به 5 بیت () نیاز داریم (قسمت index). در نهایت نیز 13 بیت می آدرس دهی می ماند که مربوط می شود به بخش tag.

| Tag (13bits) | Index (5bits) | Offset (1bit) |
| --- | --- | --- |

## **(تمرین) سوال چهارم (تمرین سوم نیم سال دوم 00-99 دکتر زرندی - با کمی تغییر)** برنامه ای را در نظر بگیرید که درخواست های دسترسی به حافظه ی آن مانند تصویر زیر باشد (هرجا کلمه ی branch نوشته شده، یعنی دو آدرس از نظر فیزیکی در حافظه ی اصلی، پشت سر هم نیستند).



حال تصور کنید که قرار است این برنامه را روی سیستمی با ویژگی های زیر اجرا کنید :

- دارای حافظه ی نهان با 4 بلاک (هر بلاک، یک کلمه است)

- دارای یک حافظه ی اصلی با 256 بلاک

- برای نگاشت خانه های حافظه به حافظه ی نهان از نگاشت مستقیم استفاده شده است.

پس از اجرای کامل این برنامه، نرخ موفقیت (hit rate) حافظ هی نهان سیستم چند خواهد بود ؟

**پاسخ**

برای حل این سوال باید بلاک های فراخوانی شده را به ترتیب در نظر بگیریم و وضعیت hit یا miss آنها را بشماریم. ابتدا وضعیت فراخوانی ها برای یک دور لوپ بزرگ پنج تایی را بررسی می کنیم :

Address : 160, 161, 89, 90, 89, 90, …, 91, 219

Cache block : 0, 1, 1, 2, 1, 2, …, 3, 3

hit/miss : m, m, m, m, h, h, …, m, m

Hits : 18, Total : 24

در شروع کار که هیچکدام از داده ها داخل حافظه نهان وجود ندارند پس همه miss می شوند. وقتی به آدرس 89 و 90 می رسیم و پس از miss شدن در حافظه نهان ذخیره می کنیم، 9 بار دیگر درخواست این دو آدرس تکرار می شود در نتیجه در اینجه 9 \* 2 = 18 hit رخ می دهد. در نهایت نیز برای آدرس های 91 و 219 درخواست داده می شود که هر دو miss می شوند. پس از یک دور کامل وضعیت حافظه نهان به این صورت است :

Cache Block Address

0 160

1 89

2 90

3 219

در 4 دور بعدی لوپ بزرگ 5 تایی، آدرس 160 hit می شود. سپس چون در بلاک 1 حافظه نهان مقدار ادرس 89 وجود دارد درخواست آدرس 161 miss می شود و چون دوباره مقدار آدرس 161 در آن قرار می گیرد 89 نیز miss می شود. اما مقدار ادرس 90 را در بلاک 2 داریم پس 90 hit می شود و پس از آن نیز 9 \* 2 = 18 hit داریم. درخواست آدرس های 91 و 219 هم هردو miss می شوند.

Address : 160, 161, 89, 90, 89, 90, …, 91, 219

Cache block : 0, 1, 1, 2, 1, 2, …, 3, 3

hit/miss : h, m, m, h, h, h, …, m, m

Hits : 20, Total : 24

در نتیجه تعداد کل hit ها برابر می شود با : 20\*4 + 18 = 98

Hit Rate = 98/(24\*5) = 0.8166

### **(کوییز یا میانترم ورژن کوتاه) سوال پنجم (سوال تیپیک این مبحث با اعداد رندم)** در یک حافظه نهان با سیاست جایدهی نگاشت مستقیم و با چهار بلوک، محتوای آدرس های زیر به ترتیب از چپ به راست درخواست شده است. وضعیت موفقیت هریک از درخواست ها و نرخ موفقیت را مشخص کنید. ([برای تبدیل اعداد هگز به دسیمال](https://www.rapidtables.com/convert/number/how-hex-to-decimal.html))

6A, 6B, 6C, 2C, 2D, 6A, 6B, 6C, 2C, 2D, 11, D, 41, 6B, 11, 3D, 13, 2B, 3D, 2D

**پاسخ**

| 2D | 2C | 6C | 6B | 6A | 2D | 2C | 6C | 6B | 6A | **Hex** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 45 | 44 | 108 | 107 | 106 | 45 | 44 | 108 | 107 | 106 | **Dec** |
| 1 | 0 | 0 | 3 | 2 | 1 | 0 | 0 | 3 | 2 | **Block** |
| h | m | m | h | h | m | m | m | m | m | **h/m** |

| 2D | 3D | 2B | 13 | 3D | 11 | 6B | 41 | D | 11 | **Hex** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 45 | 61 | 43 | 19 | 61 | 17 | 107 | 65 | 13 | 17 | **Dec** |
| 1 | 1 | 3 | 3 | 1 | 1 | 3 | 1 | 1 | 1 | **Block** |
| m | h | m | m | m | m | h | m | m | m | **h/m** |

Hits : 5, Total : 20

Hit Rate = 0.25

### **(تمرین ترکیب با سوال شایان) سوال ششم (اسلاید 4 - نمونه کد از تمرین سوم نیم سال دوم 00-99 دکتر زرندی)** در یک سناریو (می توانید قطعه کد آن را بنویسید) مثالی از همجواری زمانی و همجواری مکانی بیاورید و توضیح دهید که چگونه حافظه نهان از این ویژگی ها بهره می برد.

**پاسخ**

در ساختار آرایه، درایه های آن در حافظه در خانه های مجاور هم قرار می گیرند و این نمونه ای از همجواری مکانی است. همین طور در برنامه نویسی یکی از مواردی که بسیار مورد استفاده قرار می گیرد حلقه ها (loop) هستند که موجب می شوند از برخی خانه های حافظه چندین دفعه پشت سر هم استفاده شود. به این حالت همجواری زمانی گفته می شود.

ذخیره کردن یک بلوک داخل حافظه نهان موجب می شود تا ورد (word) های مجاور هم، همزمان داخل حافظه نهان ذخیره شوند تا در صورتی که در درخواست های بعدی که احتمال زیادی وجود دارد از آنها درخواست شود، در حافظه نهان موجود باشد (مجاورت مکانی). حال با توجه به اینکه بلاک داده های استفاده شده در حافظه نهان ذخیره می شود، درصورت درخواست دوباره آنها، می توان آنها را در حافظه نهان پیدا کرد و به اصطلاح hit می شوند (مجاورت زمانی).

SUM(int\* v, int n){

int i, sum = 0

for(i=0; i<n; i++)

sum+= v[i]

}

**سوال هفتم (اسلاید 6)**توضیح دهید که افزایش اندازه بلوک های حافظه، چه مزیت و چه معایبی می تواند داشته باشد.

**پاسخ**

مزیت :

با افزایش اندازه هر بلوک با توجه به مجاورت مکانی، احتمال اینکه داده های درخواست شده بعدی در آن بلوک موجود باشند افزایش می یابد و hit rate زیاد می شود.

معایب :   
اولا در صورتی که اندازه حافظه نهان ثابت باشد با افزایش سایز بلوک ها رقابت سر مکان ها در حافظه نهان بیشتر می شود. یعنی با توجه به اینکه با افزایش اندازه بلوک ها تعداد بلوک کمتری در حافظه نهان جا می گیرد احتمال اینکه در یک جایگاه خاص چندین بار داده لود شود و داده قبلی را دور بریزد و جایگاه های دیگر خالی بمانند بالا می رود.  
دوما به اصطلاح pollution افزایش می یابد یعنی اینکه مجاورت مکانی تا حدی وجود دارد و اگر اندازه بلوک از حدی بیشتر شود عملا داده های ذخیره شده در حافظه نهان شامل مجاورت مکانی نمی شوند و ذخیره آنها بیهوده است.  
از دیگر معایب می توان به افزایش miss penalty اشاره کرد. Miss penalty به هزینه انتقال داده از main memory به حافظه نهان گفته می شود که در هنگام miss شدن رخ می دهد. واضح است که با افزایش اندازه بلوک ها، به این علت که هر دفعه حجم داده بیشتری از main memory به حافظه نهان باید انتقال یابد، هزینه این انتقال افزایش می یابد به طوری که می تواند در مواردی مزیت افزایش hit rate را خنثی کند.

**سوال هشتم (اسلاید 6)**الف - سیاست write-back و بلوک dirty را توضیح دهید.

ب - write-miss چیست و سیاست write-back با چه روش هایی به آن پاسخ می دهد ؟

**پاسخ**

الف - در این سیاست به جای اینکه داده هایی که قرار است در حافظه اصلی نوشته شوند را CPU مستقیما یا با استفاده از buffer در آن بنویسد، ابتدا فقط در حافظه نهان نوشته می شوند. به بلوکی در حافظه نهان که یکی از word های آن تغییر کرده (در واقع توسط CPU بازنویسی شده) بلوک dirty گفته می شود. در این سیاست، وقتی که درخواست یک block داده از cache می شود، اگر miss رخ دهد و اگر بلوکی که در خانه ای از حافظه نهان قرار دارد که بلوک جدید می خواهد جایگزین آن شود (بلوکی که قرار است از حافظه نهان پاک شود) بلوک dirty باشد، قبل از آنکه بلوک جدید جایگزین بلوک قدیمی شود آن بلوک قدیمی باید در حافظه اصلی write شود. برای اینکه در این فرایند حافظه نهان معطلی اضافه نداشته باشد این فرایند توسط یک buffer انجام می شود، یعنی ابتدا بلوک در یک buffer که نزدیک حافظه نهان است و نوشتن در آن سریع انجام می شود، نوشته می شود و پس از آن که حافظه نهان به کار های بعدی خود رسیدگی می کند buffer آن داده را داخل حافظه اصلی ذخیره می کند.

ب - هنگامی که بخواهیم داده ای را در حافظه write کنیم ابتدا باید بلوک حاوی آن word را داشته باشیم، word مورد نظر را آپدیت کنیم و سپس آن را به حافظه اصلی منتقل کنیم زیرا واحد ارتباط حافظه اصلی و حافظه نهان بلوک ها هستند و نمی توان یک word را مستقیما از حافظه نهان به حافظه اصلی نوشت. در صورتی که این بلوک داخل حافظه نهان وجود نداشته باشد write-miss رخ می دهد، از آنجایی که CPU نیز دسترسیی به آن ندارد، ابتدا باید این بلوک از حافظه اصلی خوانده شود و روی حافظه نهان نوشته شود، word مورد نظر در آن بازنویسی شود و پس از آن، آن بلوک دوباره به حافظه اصلی منتقل شود. البته روش دیگری هم وجود دارد که اصلا بلوک داخل حافظه نهان آپدیت نشود و word مستقیما در حافظه اصلی نوشته شود اما معمولا در سیاست write-back از روش اول استفاده می کنند.

**سوال نهم (اسلاید 7)**فرض کنید حافظه نهانی داریم که به اندازه 4 بلوک جا دارد. محتوای آدرس های زیر به ترتیب از چپ به راست درخواست شده است، وضعیت موفقیت آنها و نرخ موفقیت را برای هر یک از سیاست های جایدهی زیر بدست آورید. ([برای تبدیل اعداد هگز به دسیمال](https://www.rapidtables.com/convert/number/how-hex-to-decimal.html))

3A, 4C, 13, 3B, 11, 3A, 4C, 12, 45, A3, D9, D8, 45, 12, B2, 45, 17, D8, D9, A3

* Direct Mapping
* 2 - way set associative (with FIFO)
* Fully associative (with LRU)

**پاسخ**

Direct mapping :

| Hex | 3A | 4C | 13 | 3B | 11 | 3A | 4C | 12 | 45 | A3 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Dec | 58 | 76 | 19 | 59 | 17 | 58 | 76 | 18 | 69 | 58 |
| Block | 2 | 0 | 3 | 3 | 1 | 2 | 0 | 2 | 1 | 2 |
| h/m | m | m | m | m | m | m | h | m | m | m |

| Hex | D9 | D8 | 45 | 12 | B2 | 45 | 17 | D8 | D9 | A3 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Dec | 217 | 216 | 69 | 18 | 178 | 69 | 23 | 216 | 217 | 58 |
| Block | 1 | 0 | 1 | 2 | 2 | 1 | 3 | 0 | 1 | 2 |
| h/m | m | m | m | m | m | h | m | h | m | m |

Hits : 3 , Total : 20

Hit rate : 0.15

2 - way set associative (with FIFO) :

| Hex | 3A | 4C | 13 | 3B | 11 | 3A | 4C | 12 | 45 | A3 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Dec | 58 | 76 | 19 | 59 | 17 | 58 | 76 | 18 | 69 | 58 |
| Set | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
|  | 3A | 4C | 13 | 3B | 11 | 4C | 4C | 12 | 45 | 3A |
|  | - | 3A | - | 13 | 3B | 3A | 3A | 4C | 11 | 12 |
| h/m | m | m | m | m | m | h | h | m | m | m |

| Hex | D9 | D8 | 45 | 12 | B2 | 45 | 17 | D8 | D9 | A3 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Dec | 217 | 216 | 69 | 18 | 178 | 69 | 23 | 216 | 217 | 58 |
| Set | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
|  | D9 | D8 | D9 | 12 | B2 | D9 | 17 | D8 | 17 | A3 |
|  | 45 | 3A | 45 | D8 | 12 | 45 | D9 | B2 | D9 | D8 |
| h/m | m | m | h | m | m | h | m | m | h | m |

Hits : 5 , Total : 20

Hit rate : 0.25

Fully associative (with LRU) :

| Hex | 3A | 4C | 13 | 3B | 11 | 3A | 4C | 12 | 45 | A3 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 3A | 4C | 13 | 3B | 11 | 3A | 4C | 12 | 45 | 3A |
|  | - | 3A | 4C | 13 | 3B | 11 | 3A | 4C | 12 | 45 |
|  | - | - | 3A | 4C | 13 | 3B | 11 | 3A | 4C | 12 |
|  | - | - | - | 3A | 4C | 13 | 3B | 11 | 3A | 4C |
| h/m | m | m | m | m | m | m | m | m | m | h |

| Hex | D9 | D8 | 45 | 12 | B2 | 45 | 17 | D8 | D9 | A3 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | D9 | D8 | 45 | 12 | B2 | 45 | 17 | D8 | D9 | A3 |
|  | 3A | D9 | D8 | 45 | 12 | B2 | 45 | 17 | D8 | D9 |
|  | 45 | 3A | D9 | D8 | 45 | 12 | B2 | 45 | 17 | D8 |
|  | 12 | 45 | 3A | D9 | D8 | D8 | 12 | B2 | 45 | 17 |
| h/m | m | m | h | m | m | h | m | m | m | m |

Hits : 3 , Total : 20

Hit rate : 0.15

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# Shayan

1- الف ) بخش های کنترلی و datapath یک از اجزای کامپیوتر ما را تشکیل میدهند؟

1- ب ) کامپیوتر چگونه با دنیای خارج ارتباط برقرار میکند؟ مدیریت و برنامه ریزی این عملیات ها بر عهده کیست؟

1- ج ) چرا کامپیوتر به حافظه نیاز دارد؟

لکچر (1)

# 

# الف ) بخش های کنترلی و datapath در واقع cpu پردازنده کامپیوتر ما هستند.

# ب ) اطلاعات از دنیای خارج توسط دیوایس هایی که input ما هستند ، وارد کامپیوتر میشوند و میتوانند در حافظه ذخیره شوند . در مرحله بعد از سمت حافظه اطلاعات خوانده میشود و دستورات و برنامه نوشته و یا داده ها به سمت پردازنده میروند و پردازش روی آن انجام میشود و بعد از آن به حافظه برمیگردد و در نهایت به بخش خروجی منتقل میشود و نتیجه نشان داده میشود و واحد کنترلی تمام این تراکنش ها را مدیریت و برنامه ریزی میکند.

# ج) چون کامپیوتر یک ماشین برنامه پذیر است و برای اینکه بتوانیم به آن برنامه بدهیم ،لازم است که این برنامه و مجموعه دستورات و داده ها در مکانی ذخیره کنیم در نتیجه به یک المانی مانند حافظه نیاز داریم.

### 2-(تمرین) آیا ویژگی های در معماری فن نیومن وجود دارد که موجب کاهش سرعت شود؟ آیا می توانید راه حل هایی برای هر یک از این ویژگی ها بیان کنید؟

لکچر(2)

در این معماری یک گلوگاه حافظه داریم ، به این صورت که برای خواندن داده و دستورات تنها یک گذرگاه داده داریم در نتیجه به طور همزمان نمیتوان هم داده و دستورات را خواند و همچنین نمیتوان عمل خواندن و نوشتن از حافظه را به شکل همزمان انجام داد. همچنین در مدل فون نیومن مراحل پردازش دستورالعمل به شکل متوالی انجام می شود در صورتی که این مراحل برای دو دستورالعمل مختلف میتواند به شکل موازی انجام شود چون دو دستورالعمل از هم مستقل هستند.

ب) امتیازی

برای مورد اول راه حلی که میتوان ارائه داد این است که داده و دستورات را در مکان هایی جدا ذخیره کنیم و برای هریک گذرگاه های داده جدا بگذاریم که این ویژگی در معماری هاروارد که معماری مدرن تری است دیده میشود هرچند که ایراد این مدل نیز این است که هزینه بیشتری میبرد. برای جبران کاهش سرعت رخ داده در اجرای متوالی هم میتوانیم از ساختارهای خط لوله (pipeline) ها استفاده کنیم تا بتوانیم تا حدودی اجرای بخش های مختلفی از دو دستورالعمل متفاوت را به شکل موازی پیش ببریم تا افزایش سرعت داشته باشیم

## 4- (تمرین) فرض کنید بخش های a , b , c روی پردازش یک برنامه کار میکنند به گونه ای که a ، چهل درصد ابتدایی پردازش را انجام میدهد و کار را تحویل b میدهد و 25 درصد از پردازش را b انجام میدهد و تحویل c میدهد و باقی پردازش را c انجام داده و کار به اتمام میرسد . اگر سرعت b ، بیست درصد افزایش یابد و سرعت c ، سی درصد افزایش یابد ، speedUp کل را به دست آورید. حال اگر بخواهیم speedup را به 1.3 برسانیم ، سرعت کار a باید چند درصد افزایش یابد

اگر دقت کنیم ، این مسئله مشابه مسائل دیگری است که با قانون amdahl حل میشود و صرفا صورت سوال تغییر کرده است.

1 / (0.4 + (0.25 / 1.2) + (0.35 / 1.4)) = 1.165

برای بخش دوم میخواهیم speedup را به 1.3 برسانیم در نتیجه داریم :

1 / ( (0.4 / x) +(0.25 / 1.2) + (0.35 / 1.4)) = 1.3 -> 0.4 / x = 0.31 -> x = 1.29

در نتیجه بخش a باید سرعت خود را 29 درصد افزایش دهد

۵.(کوییز) به سوالات زیر به صورت کوتاه پاسخ دهید.

الف) حافظه ی دائمی ای که داده ها و دستورالعملهای راه اندازی کامپیوتر را در خود جای داده , داده ها پس از خاموش

شدن پاک نمیشوند،چیست؟

ب) کدام حافظه باید بارها در ثانیه تازه سازی شود؟

ج) کدام حافظه پس از قطع شدن منبع انرژی اطلاعاتش پاک میشود؟

حذف شود:

د) تفاوت حافظه ی دسترسی تصادفی و حافظه ی دسترسی ترتیبی چیست؟ هرکدام چه مزایا و معایبی دارند؟یک

کاربرد از هرکدام را در کامپیوتر بنویسید.

لکچر (4) - سوال تمرین استاد زرندی

الف) ROM

ب) DRAM

ج) RAM

د) حافظه دسترسی تصادفی یا RAM به داده های ذخیره شده اجازه میدهد تا مستقیما ً در هر مرحله تصادفی در

دسترس باشد اما حافظه هایی مثل هارد دیسکها و...که جزو حافظه های دسترسی ترتیبی هستند،اطلاعات را به طور

متوالی در مراحل از پیش تعیین شده میخوانند و ثبت میکنند.بنابراین زمان دسترسی به داده ها به مکان آن بستگی

دارد.از RAM در حافظه CPU cache استفاده میشود.

RAM یک حافظه موقت برای ذخیره سازی داده ها برای برنامه هایی که نیاز به دسترسی سریع و مداوم به داده ها دارند

اما فرار است و معمولا با سایر حافظه ها در ارتباط است

### 6-(میانترم، کوییز، تمرین) الف) مزیت EEPROM به نسبت EPROM چیست ؟

6- ب) تفاوت ROM و PROM و را شرح دهید ؟

لکچر (4)

الف) از آنجایی که عملیات erase کردن در EEPROM به شکل الکتریکی صورت میگیرد در نتیجه نیازی نیست این حافظه را جهت بردن به حالت خام از مدار خارج کنیم تا زیر نور فرابنفش قرار دهیم که پاک شود و با عبور جریان الکتریکی می توان آن را erase کرد.

ب) PROM به عنوان یک حافظه خالی تولید میشود ، به گونه ای که تمام بیت های آن به وسیله اتصال دیود ، یک میشود و میتوان آن را بعدا برای یک بار برنامه ریزی کرد اما ROM در طول فرایند تولیدش از پیش برنامه ریزی میشود و دیگر قابلیت برنامه ریزی ندارد

## 7-(تمرین) اصل مجاورت و انواع آن را توضیح دهید.

## قطعه کدی بنویسید که انواع مجاورت در آن موجود باشد و رخ دادن مجاورت‌ها در کد را مختصرا توضیح دهید.

## اصل مجاورت چه بهره ای برای ما دارد؟

لکچر (4)

اصل مجاورت میگوید که برنامه ها در زمان اجرا در هر لحظه با بخش کوچکی از فضای آدرس خودشان کار دارند.

انواع :

مجاورت زمانی : خانه هایی از حافظه که اخیراً به آنها دسترسی پیدا شده است احتمالاً به زودی دوباره مورد استفاده و دسترسی قرار می گیرند مانند اجرای حلقه ها یا استفاده از متغیرها

مجاورت مکانی : خانه هایی نزدیک به خانه های حافظه که اخیراً به آنها دسترسی پیدا کرده ایم احتمالاً به زودی مورد دسترسی قرار می گیرند مانند خانه های آرایه یا دسترسی به دستورات متوالی

بر مبنای اصل مجاورت ما میتوانیم سلسله مراتب حافظه را داشته باشیم تا افزایش سرعت را داشته باشیم به گونه ای که همه چیز را روی دیسک ذخیره میکنیم و مواردی را از دیسک که اخیراً به آنها دسترسی پیدا شده است (و در نزدیکی هم قرار دارند) در حافظه DRAM کوچکتری کپی میکنیم (Main Memory)

و مواردی را از Main Memory که اخیراً به آنها دسترسی پیدا کرده اید (و در نزدیکی هم قرار دارند ) ، در حافظه SRAM کوچکتر کپی میکنیم ( حافظه کش متصل به CPU )

این مسئله باعث افزایش سرعت در دسترسی و کاهش هزینه در به کارگیری حافظه های گران قیمت می شود.

SUM(int\* v, int n){

int i, sum = 0

for(i=0; i<n; i++)

sum+= v[i]

}

# 

# Javad- lec 2, 3

## 1 - (تمرین) رابطه بین throughput و latency را در نظر بگیرید:

* در چه صورتی میتوان گفت که این دو با یکدیگر رابطه عکس دارند؟
* اگر throughput برای سیستمی افزایش یافته باشد آیا میتوان نتیجه گرفت که latency تسک ها کاهش یافته است؟ توضیح دهید.
* - (تمرین ترکیب با سوال جواد یامیان ترم یا کوییز) دو خط تولید آب آشامیدنی داریم . در خط تولید اول تعداد 2500 بطری 1.5 لیتری آب آشامیدنی در 6 ساعت بسته بندی میشود. در خط تولید دوم تعداد 10000 بطری 4 لیتری آب آشامیدنی در 48 ساعت بسته بندی میشود . شاخصی جهت مقایسه throughput این دو خط تولید ارائه کرده و throughput این دو خط را با یکدیگر مقایسه کنید.

سوال از متن اسلاید و کلاس، جواب:

الف)‌ در صورتی که یک پردازش گر single-core داشته باشیم و پردازش موازی صورت نگیرد میتوان رابطه عکس برای این دو مورد در نظر گرفت.

ب)‌ لزوما نه. همانطور که در بخش الف گفت تنها در صورتی عبارت فوق صحیح است که مواردی مثل پردازش موازی نداشتیم و single-core باشیم. اما اگر به سراغ پردازش موازی برویم و چندین core داشته باشیم ممکن است حتی با افزایش latency هم شاید افزایش throughput باشیم.

ج) میتوانیم throughput را برای این مسئله به شکل زیر تعریف کنیم:

تعداد بطری تولید شده در یک روز \* حجم هر بطری = throughput

خط تولید اول در 6 ساعت 2500 بطری بسته بندی میکند در نتیجه در 24 ساعت 10000 بطری بسته بندی میکند

خط تولید دوم در 48 ساعت 10000 بطری بسته بندی میکند در نتیجه در 24 ساعت 5000 بطری بسته بندی میکند

حال مقایسه را انجام میدهیم

خط تولید اول = 10000 \* 1.5 = 15000 || خط تولید دوم = 5000 \* 4 = 20000

در نتیجه خط تولید دوم با وجود اینکه تعداد بطری کمتری را بسته بندی میکند اما از جهت حجم آب معدنی در نهایت throughput بالاتری دارد.

## 2- (تمرین) دنباله دستورات اجرا شده را در انواع ماشین های Accumulator ، Register-Memory و Load-Store برای اجرای دستور D= A + B - C بنویسید .

سوال از نمونه سوال ترم پیش استاد فربه، جواب:

If: A+B -> E

| Accumulator | Register-Memory | Load-Store |
| --- | --- | --- |
| Load A  Add B  (store E, load E)  Sub C  Store D | Add A, B, E  Sub E, C, D | Load R1, A  Load R2, B  Add R3, R1, R2  Load R4, C  Sub R5, R3, R4  Store R5 |

### 3- (تمرین) response time را تعریف کنید و توضیح دهید که زا چه بخش‌هایی تشکیل شده است ( برای محاسبه آن چه مواردی در نظر گرفته میشود)

سوال از متن اسلاید و کلاس، جواب:

* Total time to complete a task
* OR: how long it takes to do a task
  + Processing, I/O, OS overhead, idle time
* Response time consists of:
  + CPU time
    - CPU time spent on a program
    - The amount of time used for processing instructions
  + I/O time
    - Time elapsed to wait for I/O operations

### 4- (کوییز سوالا محاسباتی یا مفهومی، چند قسمت) فرمول محاسبه CPU time از سه پارامتر تشکیل شده است این فرمول را بنویسید موارد موثر بر هر پارامتر را نام ببرید.

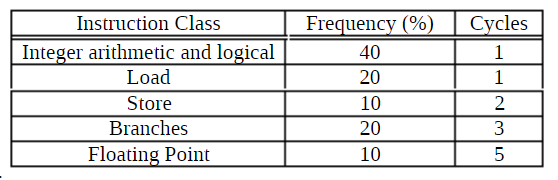
سوال از متن اسلاید و کلاس، جواب:

CPU time = IC \* CPI \* T = IC \* CPI / F

* Instruction Count
  + Program (algorithm, language)
  + ISA
  + Compiler
* CPI
  + uArch: the way processor is implemented
  + Code CPI also depends on the program
* Clock Cycle Time (or CPU Frequency)
  + Technology and uArch

## 5-(تمرین)

## الف) فرض کنیم که شما یک برنامه را روی پردازنده MHz 300 خود اجرا کرده اید. با استفاده از جدول زیر CPI و MIPS را محاسبه کنید. [منبع](https://www.docsity.com/en/amdahl-s-law-computer-architecture-and-engineering-solved-exams/298659/)



ب) فرض کنید که بهینه سازی اعمال کرده اید که 30٪ از دستورالعمل های arithmetic/logic (یعنی 12٪ از کل دستورالعمل ها)، 30٪ دستورالعمل های load و 20٪ از دستورالعمل های floating-point را حذف می کند. Speed up برنامه بهینه شده چقدر است؟ (محسبات را به طور کامل توضیح دهید)

جواب:

درحالت عادی برای محاسبه CPI صرفا کافی بود از تعداد کلاک‌های لازم برای هر نوع instruction میانگین بگیریم اما چون در اینجا فرکانس instruction ها با یکدیگر متفاوت است برای محاسبه CPI باید میانگین وزن داری از تعداد کلاک تمام instruction class ها بگیریم. پس داریم:

CPI = (40 \* 1) + (20 \* 1) + (10 \* 2) + (20 \* 3) + (10 \* 5)/100 = 1.9

برای محاسبه MIPS هم کافیست فرکانس کاری پردازنده را بر CPI تقسیم کنیم چرا که فرکانس در این سوال برحسب Ghz بیان شده است. پس داریم:

MIPS = 300/1.9 = 157.9

ب) فرمول پایه speedup مطابق زیر است

speedup =

روش ساده تر برای حل این سوال این است که ببینیم با این بهینه سازی CPI جدید چخ تغییری دارد پس داریم:

New CPI = 1.9 - [ (0.3 \* 0.4 \* 1) + (0.3 \* 0.2 \* 1) + (0.2 \* 0.1 \* 5) ] = 1.62

speedup = = = 1.17

### 6- (میانترم با تغییر یا تمرین)باتوجه به جدول زیر به سوالات پاسخ دهید. (زمان اجرا را برابر CPU time درنظر بگیرید )

| **Processor C** | **Processor B** | **Processor A** |  |
| --- | --- | --- | --- |
| **2.5** | **3** | **4** | **Frequency (Ghz)** |
| **1.5** | **2** | **1.5** | **CPI** |

الف) پردازنده ها را براساس معیار MIPS مرتب کنید.

ب) برنامه p توسط هر یک از این پردازنده ها در مدت زمان 30 ثانیه اجرا میشود. آیا میتوانید تعداد کلاک و تعداد instructionهای اجرا شده در برای هر پردازنده را محاسبه کنید؟ (دقت کنید که هر P برای هر پردازنده یه طور متفاوت کامپایل شده و در هر پردازنده میتواند از منظر تعداد دستورات متفاوت باشد)

ج) قصد داریم که زمان اجرای برنامه را برای این پردازنده‌ َA را بدون اینکه CPI را برای آن تغییر دهیم به اندازه 20% کاهش دهیم. فرکانس کاری این پردازنده با به چه عددی برسد؟ ( برای IC میتوانید از مقدار بدست آمده در قسمت ب استفاده کنید)

نمونه سوال دکتر فربه همراه با تغییرات

اگر انتخاب شد جوابشو مینویسم ولی کلا فرمول اصلی حل سوال اینه:

Cpu time = (IC \* CPI)/f

#of clock = IC \* CPI

### 7- (کوییز) فرض کنید شما تنها میتوانید 40% از کد یک برنامه را بهبود ببخشید. مشخص کنید که چگونه میتوان به speedup=1.8 رسید.

سوال طرح خوم :)

جواب: امکان پذیر نیست زیرا این مقدار از بیشنه مقدار speedup با fraction = 0.4 بیشتر است.

طبق قانون ambdahl داریم:

Maximum speedup = 1/((1 - 0.4) + (0.4/infinity)) = 1/0.6 = 10/6 = 1.66

سوال مشابه در کوییز