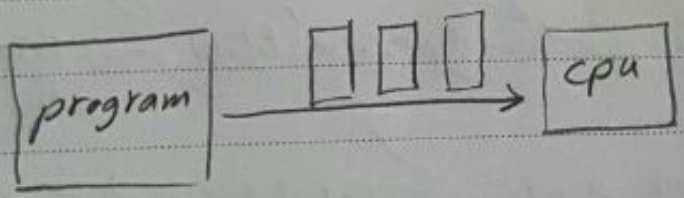
Subject :
Year . Month . Date . ( )

پردازش موازی یا محاسبه موازی به اجرای همزمان چند برنامه که به بخش های کوچکتری تقسیم شده بر روی چند پردازنده اطلاق می شود .

در واقع چند شده به زیر وظایفی تقسیم می شود که به اجرا همزمان آن چند شدی اصلی در زمان کوتاه تری حل می شود .

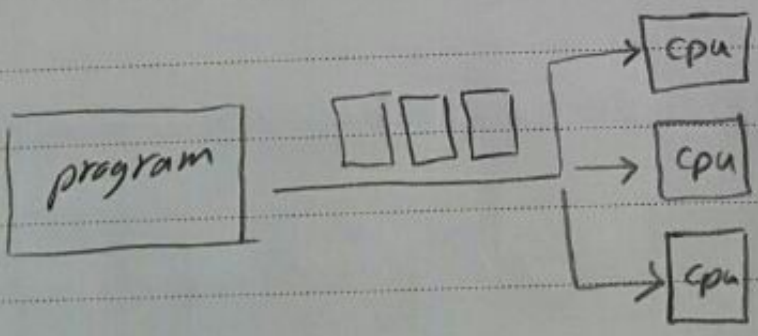
بسیاری از برنامه ها احتیاج به پردازش حجم زیادی از داده ها دارند مثل : پایگاه های عظیم داده موتورهای جستجوی وب ، تصویر برداری ، تجهیزات پزشکی ، مدل سازی های مالی و اقتصادی ، گرافیک پیشرفته ، شناسایی های چند رسانه ای ، شبکه های ویدئویی

پردازش ها



پردازش سریال

Time shaning



پردازش موازی

طبقه بندی (Flynn)

نظر به معماری کامپیوترهای موازی پیشنهاد شده است. و آنها را براساس نحوه‌ی تعامل بین دستور (instruction) و داده (data) تقسیم‌بندی می‌کند. و به چهار دسته‌ی SISD

single instruction Single data (از یک دستور - تک داده)

single instruction mutiple Data (SIMD)
multiple instruction single Data (MISD)
multiple " multiple " (MIMD)

SISD : این معماری دارای یک جریان دستوری روی یک جریان داده در طول هر پالس ساعت مورد استفاده قرار می‌گیرد. دستورات قطعی هستند و به پردازنده‌ی دیگری وابسته نیست

SIMD : یکی از انواع کامپیوترهای موازی است، همه‌ی واحدهای پردازش یک دستور مشترک را در طول هر پالس ساعت اجرا می‌کنند. اما هر cpu می‌تواند روی چند عنصر داده مختلف عمل کند

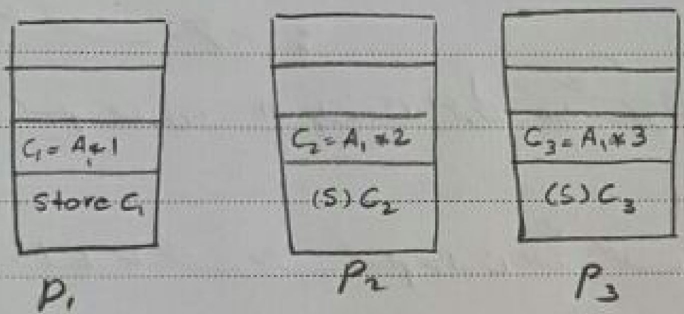
Subject:
Year. Month. Date. ( )

(MISD) : در این طرح موازی یک جریان داده به چند واحد پردازش داده ارسال می‌شود

هر واحد پردازش به طور مستقل با جریان های دستور مستقل روی دیتا عمل می‌کنند. تاکنون تعداد

معدودی کامپیوتر موازی با این روش ساخته شده.

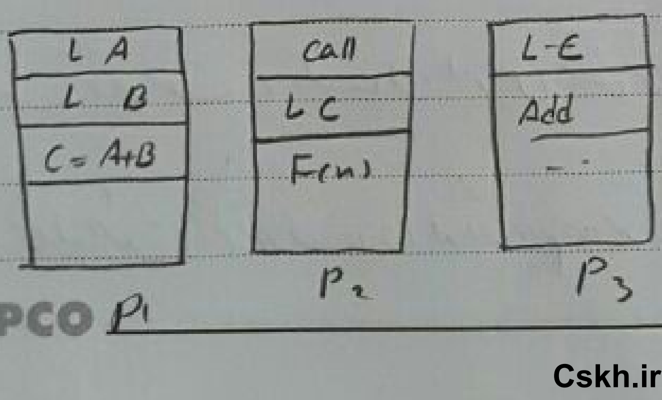
از جمله کاربردهای این روش اعمال چند الگوریتم رمزگشایی بر بازکردن پیغام رمز که داده شده است.



چند دستور روی دیتای واحد در چند پردازنده

نکته: مثل اینجا رمز پلیس (واحد جهان)

MIMD : معمول ترین طرح کامپیوترهای موازی است، هر پردازنده امکان اجرای چند جریان دستوری جداگانه روی چند جریان داده‌ی مختلف را دارد، عملیات اجرایی می‌توانند قطعی یا غیرقطعی باشد (عملیات انجام می‌شود ولی خروجی ممکن است به پردازش دیگری وابسته باشد)، کامپیوترهای موازی خوشه‌ای، ابرکامپیوترها، کامپیوترهای چند پردازنده‌ی SMP، و PC های چند هسته‌ای امروزی



از این معماری استفاده می‌کنند

PAPCO

نکته: سیستم های چند پردازنده بر دو دسته تقسیم می‌شوند: کاملاً موازی اگر همه پردازنده‌ها به طور یکسان بتوانند تمام دستورات سیستم عامل را اجرا کنند. به آنها سیستم چند پردازنده‌ی متقارن گفته می‌شود. اما اگر بعضی از پردازنده‌ها دارای امتیاز بیشتر یا کمتر نسبت به سایرین باشند، به آنها سیستم چند پردازنده‌ی نامتقارن گفته می‌شود.

کارایی نسبت به هزینه:

یک سیستم موازی با n پردازنده‌ی معمولی کارایی کمتری نسبت به یک پردازنده به سرعت n برابر دارد، اما ساخت یک سیستم موازی دارای قیمت پایین‌تری است.

برای برنامه‌هایی که حجم محاسبات فراوانی دارند و حجم داده‌ی محدودی دارند، زمانی هستند که می‌توان آنها را به n زیربرنامه (ریز) تقسیم کرد، محاسبه‌ی موازی محرک راه حل است.

الگوریتم ها:

نامی تصور می‌شود که محاسبات موازی، تنها محتاج به فراهم کردن سخت افزار مورد نیاز و اتصال درست آنهاست، ولیکن دستیابی به افزایش خطی سرعت، بسیار مشکل است. این مشکل ناشی از طبیعت ترتیبی بسیاری از الگوریتم‌هاست. به طوری که قسمت‌هایی از یک الگوریتم

غیرقابل موازی سازی است. برای اثبات این حقیقت قانون آمدال Amdhl

بیان و شکل داده می‌شود، فرض کنید F = 10٪ درصدی از الگوریتم که قابلیت موازی سازی را

ندارد. اما بقیه‌ی الگوریتم به طور موازی توسط N = 20 پردازنده اجرا می‌شود. برای اینکه

سرعت اجرای برنامه نسبت به زمانی که تنها روی یک پردازنده اجرا شود، 20 برابر بشود باید طبق

رابطه‌ی آمدال با ضریب افزایش می‌یابد

$$\frac{1}{F + \frac{1-F}{N}} = \frac{1}{\frac{1}{10} + \frac{1-\frac{1}{10}}{20}} \simeq 7 \rightarrow$$

سرعت 7 برابر می‌شود

نه 20 برابر که همان تعداد پردازنده‌هاست

× حجم

Subject:
Year. Month. Date. ( )

جلسه دوم : 95 , 8 / 19

قانون مور: عملکرد ریزپردازنده ها با فاکتور 2 (دو برابر) رو به رشد است.

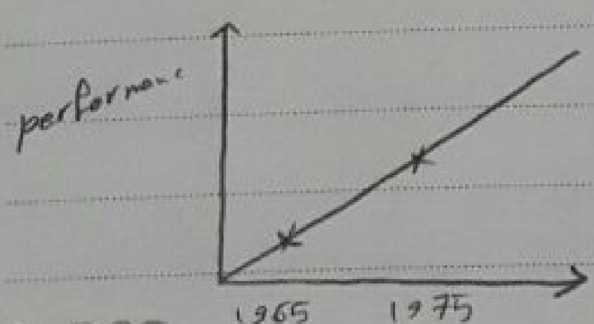
درجهت اصلاح قانون مور این زمان به 18 ماه تغییر کرد که تقریباً معادل 160٪ در سال است. این رشد ناشی از ترکیب این دو عامل است.

1- افزایش پیچیدگی تراشه (VLSI) و پیش بینی افزایش - حدود 15 میلیون ترانزیستور در یک چیپ: chip: تراشه

2- بهبود در معماری ها مانند حافظه ها cash و غیره چندنخی یا خطوط لوله عمیق، قانون مور در 1965 با مضمون دو برابر شدن پیشرفت در پیچیدگی چیپ (تراشه) در هر 18 ماه برابر با تعداد پله‌ای کم، تدوین شد. بیان قانون مور شدی IPS:

Floating per seond : FLOP , Instruction per second

عملیات ممیز شناور در ثانیه — تعداد دستورالعمل اجرایی در ثانیه توسط cpu



سرعت رشد طبق قانون مور
(کارایی cpu)

محدودیت قانون مور :

محدودیت سرعت نور ، سرعت نور 3×10^8 است اما به ترانزیستورها از سرعت نور

استفاده نمی‌شود بلکه ذرات سیگنال استفاده می‌شود که محدودیت را سرعت آن‌هاست

که همان 10^8 است .

براساس فرمول $t = \frac{v}{d}$ اگر قطر ترانزیستور 1 سانتی متر باشد .
(مسافت ، سرعت انتشار)

$$t = \frac{v}{d} = \frac{1cm}{1\times10^8} = \frac{10^{-2}}{10^8} = 10^{-10} = 0.1\times10^{-9} = 0.1\ ns$$

یعنی هر یک دهم (0.1) نانو ثانیه یک دستورالعمل اجرا می‌شود .

برای رفع این محدودیت باید مساحت یا قطر ترانزیستور را کم کرد . بنابراین باید به فکر تکنیک‌های

جدیدی سازی بود (که این کم کردن زیر 18 دیگر امکان نیست)

زیرشاخه‌های MIMD از نظریه Flynn :

شامل چند زیرکلاس

GMSV : Global memory shared variable

Subject :
Year . Month . Date . ( )

GMMP : global memory message passing

DMSV : distributed memory shared variable

DMMP : distributed memory message passing

حافظه اصلی توزیع شده بین پردازه ها تقسیم شده و روش ارتباط بین پردازه ها انتقال داده و اشکار پیام است

موانع پیشرفت حوزی :

1- قانون رانش : هزینه تحقیقات p برابر شده عامل هزینه p^2 برابر شده سیستم تک پردازنده است .

2- قانون minsky : افزایش سرعت متناسب با $p \log_2 p$ خواهد شد

3- قانون های IC : هر 5 سال سرعت 10 برابر میشود .

4- قانون آمدال : کد با دستورات غیر حوزی سرعت را به شدت محدود می کند .

نهایتاً به این نتیجه می‌رسیم که فقط موازی سازی خطا به تعداد پردازنده‌ها بستگی ندارد . بلکه به برنامه یا کدی که قرار است اجرا شود نیز بستگی دارد .

ارزیابی سیستم های پردازش موازی : به منظور ارزیابی سیستم ها به پردازش ها موازی چند سری پارامترها تعریف شده است

1 - تعداد پردازنده ها : (p)

2 - تعداد عملیاتی که پردازنده‌ی p ام انجام می‌شود : $w(p)$

3 - زمان اجرای عملیات توسط این p پردازنده : $T(p)$ ، اگر $p=1 \Rightarrow T(1)=w(1)$

4 - سرعت (از قانون آمدال تبعیت می‌کند) که از این رابطه به دست می‌آید : $S(p) = \frac{T(1)}{T(p)}$

5 - بهره‌وری : $E(p) = \frac{T(1)}{p \times T(p)}$

6 - تخت اجرایی که به صورت اضافی استفاده شده : $R(p) = \frac{w(p)}{w(1)}$

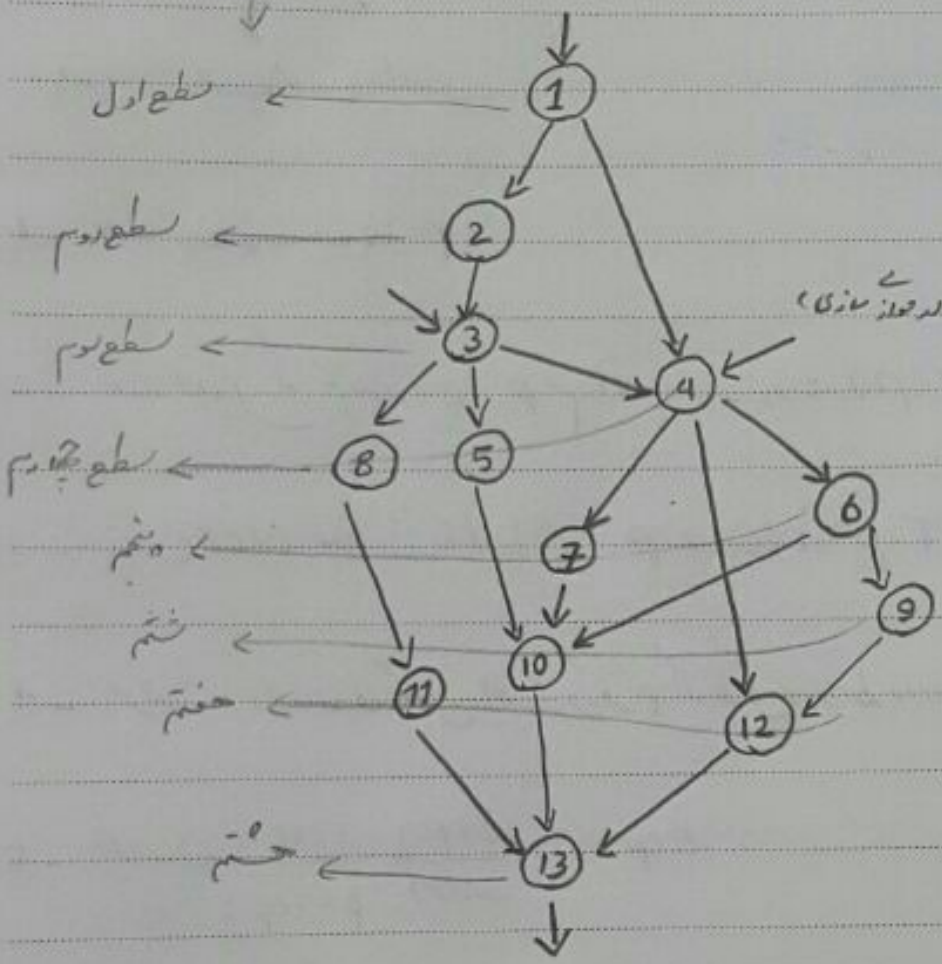
7 - کیفیت : $Q(p) = \frac{T^3(1)}{p \times T^2(p) \times w(p)}$

Subject :
Year . Month . Date . ( )

مثال : در مورد تحلیل سیستم ها

گراف زیر داده شده است اگر زمان اجرای هر پیله 1 واحد باشد پارامترهای

$W(1)$ و $T(1)$ و $T(\infty)$ را محاسبه کنید .



$W(1) = 13$

$T(1) = 13$

$T(\infty) = 8$

زمان موازی سازی نهایت 8 (سرعت در موازی سازی)

چون گراف 8 سطح است

نیز نه 8 پردازنده جواب می دهد

سطح اول ← 1

2

دوم ← 2

سوم ← 3

9, 5, 8

6, 7

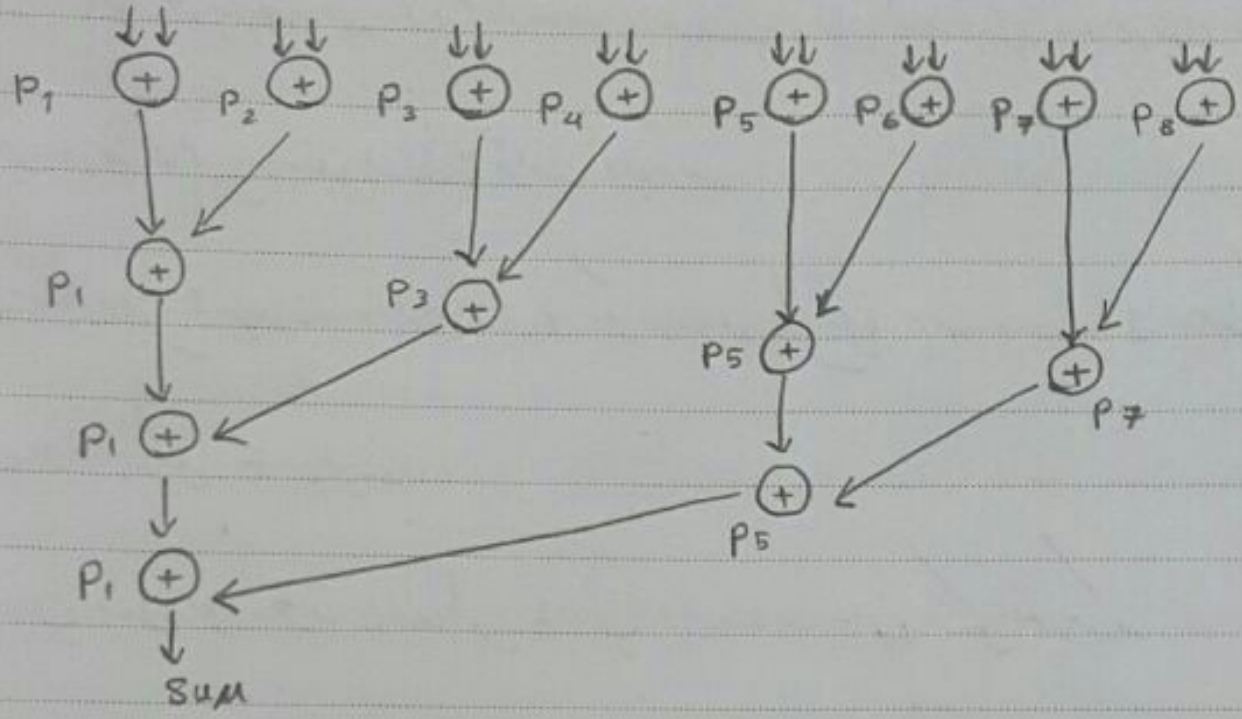
9, 10

11, 12

13

مثال دوم : جمع 16 عدد در یک سیستم 8 پردازنده ای

الف : اگر هزینه ارتباط بین پردازنده ها را 0 در نظر بگیریم ، پارامترهای $T(1)$ ، $W(1)$ ، $T(p)$ ، $W(p)$ ، $S(p)$ ، $E(p)$ ، $R(p)$ و $Q(p)$ را محاسبه کنید ؟



$T(1) = 15 \rightarrow$ تعداد عملیات ها

$T(1) = w(1) = 15$

$T(8) = 9 \rightarrow$ تعداد سطوح گراف

$w(8) = 15 \rightarrow$ (تعداد عملیات) تعداد کلی که توسط پردازنده

$$S(p) = \frac{T(1)}{T(p)} = \frac{15}{9} \simeq 3.75$$

$$E(p) = \frac{w(p)}{p \times T(p)} = \frac{15}{8 \times 9} = 0.147$$

$S(8)$

0/47 یعنی 0/53 از پردازنده ها بیکارند

$$R(p) = \frac{w(p)}{w(1)} = \frac{15}{15} \rightarrow$$ یعنی تحت افزود اضافی نیازی نیست $R=1$

$$Q(p) = \frac{T(1)}{p \times T^2(p) \times w(p)} = \frac{15^3}{8 \times 9^2 \times 15} = 1.176$$

Subject:
Year. Month. Date. ( )

نکته: اگر هزینه ارتباطی بین پردازنده‌ها در صورت سوال لحاظ نشود باید هزینه‌ی طولانی‌ترین مسیر تا نتیجه نهایی را به تعداد سطوح گراف اضافه کنید.

نکته: برای p پردازنده (به‌عنوان مثال) اگر هزینه ارتباطی بین پردازنده‌ها را 1 در نظر بگیریم این هزینه در کل 7 می‌شود.

بنابراین مثال قبل، هزینه‌ی ارتباطی 1 بین پردازنده‌ها به این شکل تغییر می‌کند

$w(p) = 15 + 7 = 22 = w(8)$ تعداد کل محاسبات + هزینه ارتباطی

$T(p) = 15 + 3$ تعداد سطوح گراف + طولانی‌ترین مسیر

$$S(p) = \frac{15}{7} = \frac{T(1)}{T(p)} = 2.14$$

$$E(p) = \frac{T(1)}{p \times T(p)} = \frac{15}{8 \times 7} = 27\%$$

$$R(p) = \frac{w(p)}{w(1)} = \frac{22}{15} = 1.47$$

$$Q(p) = \frac{T^3(1)}{p \times T^2(p) \times w(p)} = \frac{(15)^3}{8 \times (7)^2 \times 22} = 0.39$$

هزینه ارتباط همیشه در حالت سری محاسبه می‌کنیم

Subject:
Year. Month. Date. ( )

نگاهی سریع به برخی از الگوریتم های موازی :

5 نوع محاسبه ی موازی داریم :

1. محاسبات زنجیره ای semigroup زمان اجرا = تعداد سطوح = $\log_2^n$

2. محاسبات موازی پیشوندی perfix " " $\log_2^n$

3. محاسبات بسته ای packet routing

4. محاسبات انتشاری broad casting

5. مرتب سازی رکوردها sorting

نکته : محاسبات موازی مثل زنجیره ای است با این تفاوت که نتایج میانی هم نیاز است و حداقل به $\log_2^n$ مرحله برای محاسبه نیاز داریم .

معیارهای پردازنده ها :

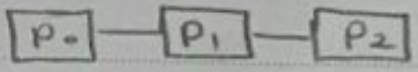
در پردازنده هایی که به صورت آرایه ای خطی به هم متصل هستند دو پارامتر تعریف می شود

D : قطر پردازنده : تفاوت بین طولانی ترین و کوتاه ترین مسیر بین یک زوج پردازنده

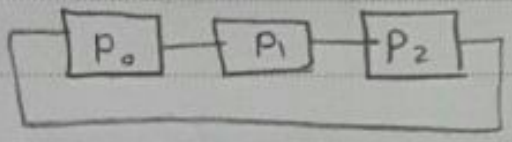
در مدل آرایه‌ی خطی معمولی D_{max} : طول مسیر $(p-1)$

حداکثر درجه هرگره (d) : حداکثر تعداد پیوندها یا کانال های ارتباطی برای یک پردازنده تعداد خط‌هایی که وارد یا خارج شده.

در آرایه خطی معمولی $d = D_{min} = 2$



در آرایه حلقوی : $d = 2$ ، $D = \lfloor P/2 \rfloor$ ⇓ 4



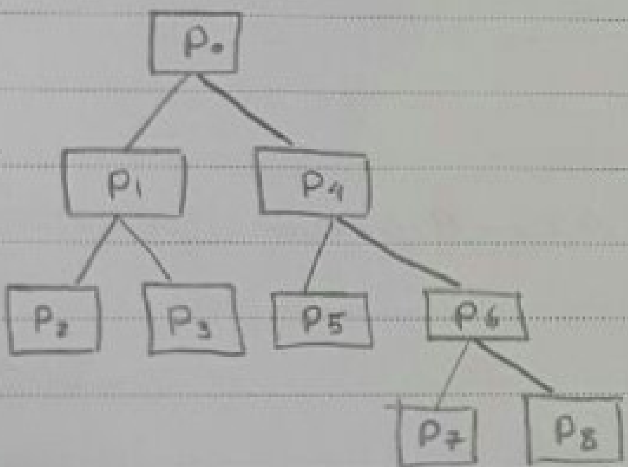
جلسه سوم : 95/9/13

پردازنده هایی که به صورت درخت دودویی به هم متصل هستند :

یک درخت باینری الزوما کامل و الزوما کاملا متوازن نمی باشد.

d (حداکثر درجه گره ها) : همانند Liner ، تعداد اشعه بت خروجی از هر نود

$D : 2\lfloor \log_2 P \rfloor$



$d : 3$

$D : 2\lfloor \log_2 9 \rfloor$

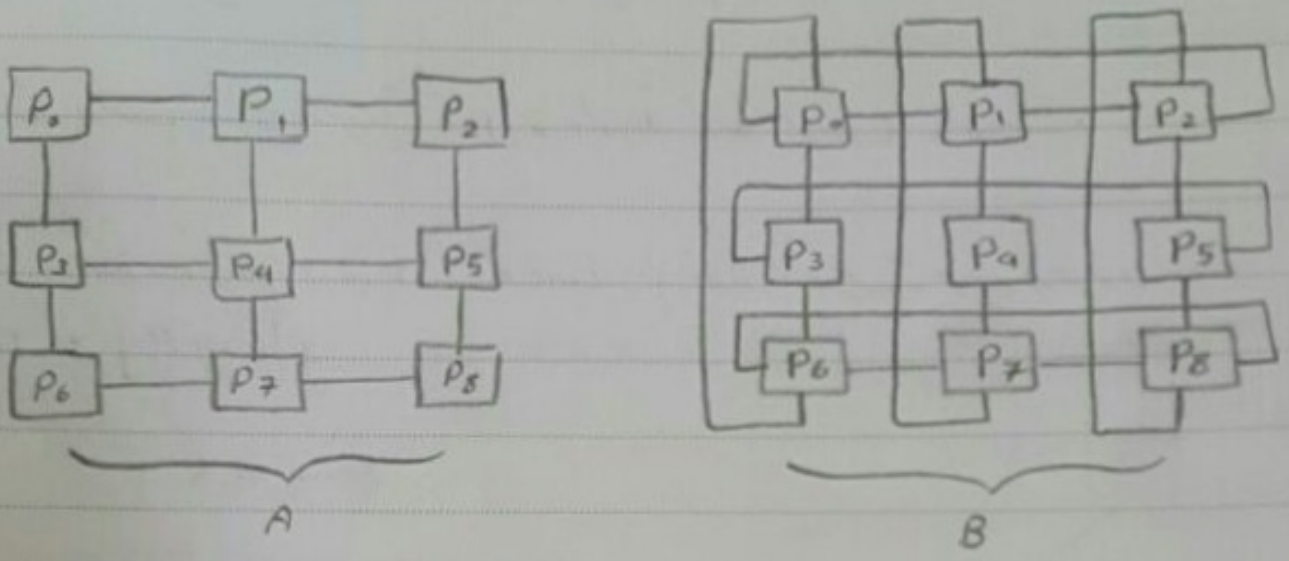
پردازنده هایی که به صورت mesh دو بعدی به هم متصل اند :

پردازنده ها به شکل مش دو بعدی می توانند یا به شکل سطر و ستون مانند شکل (A) و یا به شکل ستونی از ابتدایی ترین پردازنده به انتهای آن ستون و یا به شکل سطری (از ابتدایی ترین پردازنده به انتهای آن متصل شوند

Subject :
Year . Month . Date . ( )

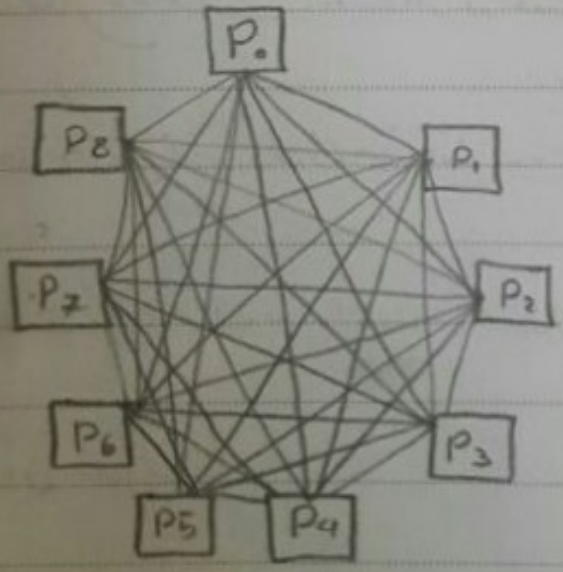
d : حداکثر درجه نودها

$D_A : 2\sqrt{P} - 2$ $\quad D_B : \sqrt{P}$



نکته : مزیت معماری نوع B به نوع A ، این است که مقدار پارامتر D کاهش یافته است .

برای نودهایی که در صورت معماری حافظه اشتراکی به هم متصلند :



$d : 8$ حداکثر درجه گره ها

$D : 1$ چون به تمام گره ها مسیر داریم

Subject:
Year. Month. Date. ( )

ترکیب چند معماری موازی : 5 الگوریتم موازی :

۱- الگوریتم های برای معماری خطی :

الف : الگوریتم زنجیره ای برای معماری خطی (Liner)

در الگوریتم semigroup (زنجیره ای) فقط نتیجه ی پایانی مهم است ، نتایج میانی مهم نیست

مثال : فرض کنید یک معماری آرایه ی خطی با n پردازنده داریم . در هر پردازنده یک عدد ذخیره شده است ، هدف پیدا کردن max این اعداد است .

(طی الگوریتم زنجیره ای معماری خطی انجام میشود)

- هر پردازنده در هر مرحله مقدار خاکریز خود را به همسایه ی مجاور خود ارسال میکند .
- هر پردازنده با دریافت مقادیر پردازنده های مجاور خاکریز را به صورت رابطه ی زیر بروز می آورد : max(Left, own, Right)
- الگوریتم زمانی خاتمه می یابد که هر پردازنده از همسایه های خود دو عدد یکسان دریافت کند .

P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	
				3	7	9	1	9	← Initial value
5	2	8	6	7	9	9	9	9	
5	8	8	8	9	9	9	9	9	
8	8	8	8	9	9	9	9	9	
8	8	8	9	9	9	9	9	9	
8	8	9	9	9	9	9	9	9	
8	9	9	9	9	9	9	9	9	
9	9	9	9	9	9	9	9	9	→ max identified

Subject:
Year. Month. Date. ( )

ب) الگوریتم prefix به‌صورتی برای n تا عدد خاص:

در این الگوریتم جواب‌های میانی خود نیاز است.

مثال: فرض کنید n تا عدد آرایه‌ی خطی با p پردازنده داریم و در هر عدد ذخیره شده پردازنده i ام است. هدف محاسبه‌ی حاصل جمع است.

- در هر مرحله i امین پردازنده مقدار خود را با مقدار پردازنده‌ی سمت چپ خود جمع کرده و در خود ذخیره می‌کند.
- در هر مرحله فقط i امین پردازنده عمل جمع را انجام می‌دهد. و بقیه‌ی پردازنده‌ها منتظر می‌مانند تا جمع انجام شده به پایان برسد.
- عمل جمع از اولین پردازنده آغاز و در مرحله‌ی بعدی تأثیر خود را روی پردازنده‌ی بعدی به عنوان ورودی جمع می‌گذارد.
- الگوریتم در آخرین پردازنده به پایان می‌رسد.

نکته: در این مثال تعدادی به لحاظ زمان اجرا این n تا سیستم p پردازنده و n تا پردازنده‌ای وجود ندارد.

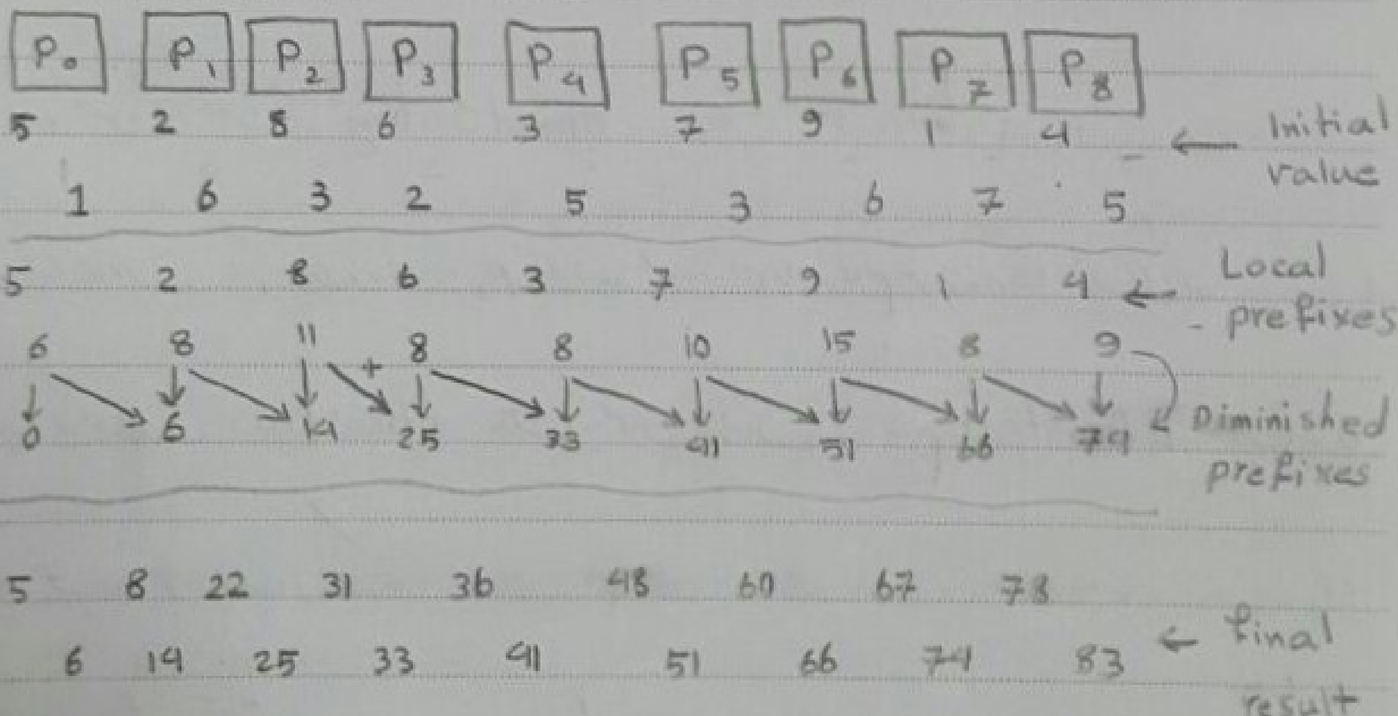
P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	
5	2	8	6	3	7	9	1	4	→ initial value
5	7	8	6	3	7	9	1	4	
5	7	15	6	3	7	9	1	4	
5	7	15	21	3	7	9	1	4	
5	7	15	21	24	7	9	1	4	
5	7	15	21	24	31	9	1	4	
5	7	15	21	24	31	40	1	4	
5	7	15	21	24	31	40	41	4	
5	7	15	21	24	31	40	41	45	→ final value

Subject:
Year. Month. Date. ( )

مثال : یک سیستم آرایه‌ای خطی با ۹ پردازنده داریم که در هر پردازنده دو عدد ذخیره شده است، هدف محاسبه‌ی حاصلجمع این اعداد است.

- محاسبه‌ی مجموعه داده‌ها در هر پردازنده انجام می‌شود.
- عملیات جمع به صورت prefix انجام می‌گیرد.
- در نهایت حاصلجمع های محاسبه شده در مراحل قبل با هم جمع می‌شود. مطابق شکل در مرحله‌ی Local .. هر پردازنده به صورت همزمان دو مقدار خود را جمع می‌کند.

در مرحله‌ی بعدی ... Diminish حاصلجمع های محاسبه شده در مرحله‌ی قبل به صورت سری و غیر همزمان با هم جمع می‌شود، در نهایت -- final ، حاصلجمع های محاسبه شده در مراحل قبلی، با یکدیگر جمع می‌شوند.

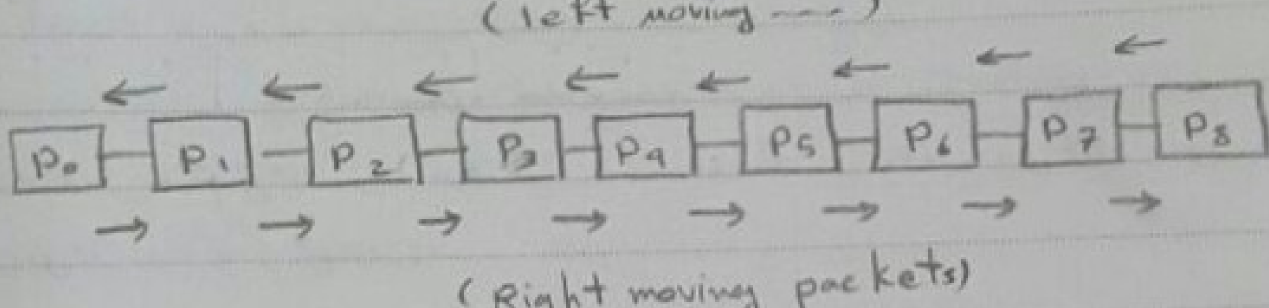


PAPCO

ج) الگوریتم مسیریابی بسته‌ای packet Routing برای (روی) آرایه خطی Liner :

پردازنده‌ی P_i می‌خواهد بسته‌ای را به P_j ارسال کند. محاسبه‌ی (j-i) برای تعیین مقصد و جهت حرکت که به آن Routing tag می‌گوییم درنظرگرفته می‌شود.

اگر این مقدار مثبت باشد حرکت به سمت راست و اگر منفی باشد حرکت به سمت چپ است



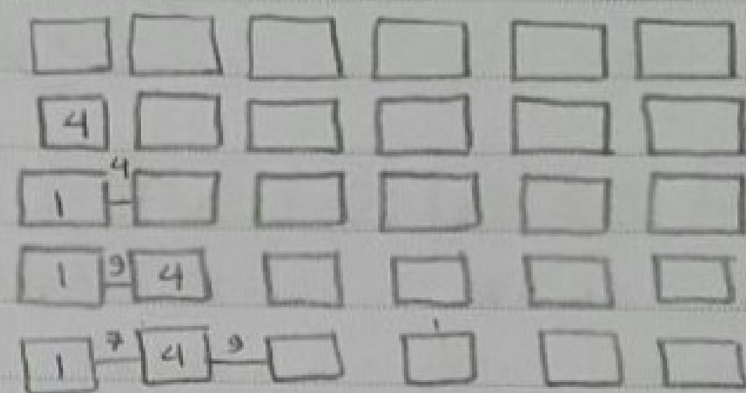
د) مسیریابی انتشار، Broadcasting روی یک معماری خطی:

در اینجا اگر پردازنده‌ی P_i بخواهد بسته‌ای را به تمام پردازنده‌ها ارسال کند. آن پیام را به صورت rbcast(a) به سمت راست و پیام lbcast(a) به سمت چپ خود ارسال می‌کند.

ه) الگوریتم مرتب‌سازی sorting روی معماری Liner : مقدار اولیه‌ی پردازنده‌ها (∞) یا بی‌نهایت است.

هر پردازنده باید مقدار جدید از سمت چپ دریافت می‌کند و آنرا با مقدار ذخیره شده در Register داخلی خود مقایسه می‌کند. مقدار کوچکتر (کوچکترین) باید داشته و بزرگتر را به پردازنده‌ی سمت راست خود انتقال می‌دهد. (الگوریتم درج)

5/2/8/6/3/7/9/1/4
5/2/8/6/3/7/9/1
5/2/8/6/3/7/9
5/2/8/6/3/7
5/2/8/6/3



ادامه تا ترتیب سازی کامل

مثال: مرتب سازی odd/Even: زوج و فرد:

های دو مرحله زوج و فرد.

• مرحله odd: در این مرحله پردازنده های فرد مقادیر خود را با همسایه های زوج سمت راست مقایسه می‌کنند. اگر مقادیر فوق مرتب نباشند، داده های خود را جابجا می‌کنند.

• مرحله Even number: در این مرحله پردازنده‌های زوج مقادیر خود را با همسایه‌های خود سمت راست مقایسه می‌کنند. اگر مقادیر مرتب نباشند، داده‌های خود را جابجا می‌کنند.

مثلاً در مرحله‌ی odd number، ابتدا به طور همزمان P_1 مقدار خود را با P_2 و P_3 مقدار خود را با P_4 و P_5 با مقدار P_6 و P_7 مقدار خود را با P_8 مقایسه می‌کنند و اگر لازم باشد

مقادیر جابه‌جا می‌شود. سپس در مرحله‌ی Even number ، P_0 مقدار خود را با P_1 و P_2

مقدار خود را با P_3 و P_4 مقدار خود را با P_5 و P_6 مقدار خود را با P_7 مقایسه

می‌کند. و اگر لازم باشد مقادیر جابه‌جا می‌شود.

الگوریتم تا زمانی ادامه می‌یابد که هیچ جابه‌جایی صورت نگیرد.

P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
5	2 ⟷	8	6 ⟷	3	7 ⟷	9	1 ⟷	4
5 ⟷	2	8 ⟷	3	6 ⟷	7	9 ⟷	1	4
2	5 ⟷	3	8 ⟷	6	7 ⟷	1	9 ⟷	4
2 ⟷	3	5 ⟷	6	8 ⟷	1	7 ⟷	4	9
2	3 ⟷	5	6 ⟷	1	8 ⟷	4	7 ⟷	9
2 ⟷	3	5 ⟷	1	6 ⟷	4	8 ⟷	7	9
2	3 ⟷	1	5 ⟷	4	6 ⟷	7	8 ⟷	9
2 ⟷	1	3 ⟷	4	5 ⟷	6	7 ⟷	8	9
1	2 ⟷	3	4 ⟷	5	6 ⟷	7	8 ⟷	9

GPU : از تعداد نسبتاً زیادی پردازنده‌ی ساده و کوچک که به آن‌ها اصطلاحاً Thread

یا نخ گفته می‌شود، ساخته می‌شود. هر کدام از این نخ‌ها می‌تواند بخش ریز یک پیکسل را در یک

ثانیه ساعت پردازش کند. قبل از معرفی (CUDA) تعداد کمی از محققین استفاده از آن را

Subject:
Year. Month. Date. ( )

آغاز کردند. اما به دلیل مشکلات زیاد کار با آن متوقف شد.

مهمترین مشکل این بود که تنها راه برقراری ارتباط با کارت گرافیک از طریق نرم افزارهای خاصی

مثل direct x و opengl بود. و این واسطه ها فقط و فقط مختص پیکسل بودند، با

معرفی (CUDA) اکثر مشکلات برطرف شد، CUDA بدین معنا که دستگاه های

تحت افزاری جانبی توان پردازشی مثل CPU داشته باشند و بتوانند بدون نیاز به CPU، پردازش

را انجام دهند. در واقع با ابتکار بارهای سنگین از روی دوش CPU برداشته شد، و قدرت

پردازش GPU از CPU (CPU) بالاتر می باشد، فلسفه ی ابتکار این است که بتوان کارت گرافیک

سنگین مد (VGA) یا همان کارت گرافیک با استفاده از CPU های تعبیه شده روی آن

به صورت جزئی و با سرعتی بیشتر از CPU اصلی عملیات را انجام می دهند. البته برنامه نویس

خودش تعیین می کند که پردازش در هر مرحله توسط CPU انجام شود یا CUDA یا همان کارت گرافیک

یا CPU در مرحله ی اول با GPU در ارتباط است. داده های تحت پردازش از حافظه ی

اصلی به حافظه ی GPU کپی می شوند. دستور پردازش توسط CPU به GPU داده می شود

Year. Month. Date. ( )

برنامه هایی که صورت خطی پی در پی اجرا می شوند (روی GPU) تا زمانی از پردازنده ها (نخ

به حافظه GPU منتقل می شود. در نهایت از GPU به حافظه ی اصلی منتقل می شود.

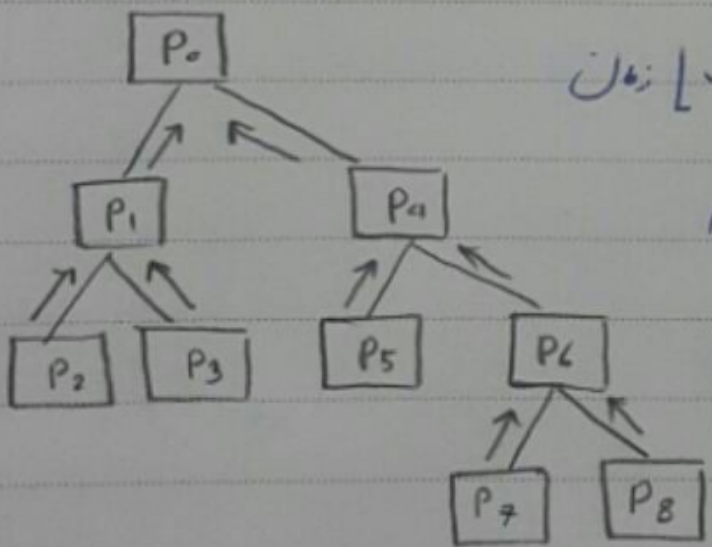
جلسه چهارم / پردازش موازی / 95/9/17

- الگوریتم های موازی روی درخت دودویی

در الگوریتم هایی که برای پردازنده ها به شکل درخت دودویی کاظم شود. فرض بر آنست که داده ها در برگ های درخت ذخیره شده اند و پردازنده های غیر برگ تنها در محاسبات شرکت می کنند اما عناصر داده را در خود نگه نمی دارند.

- محاسبات زنجره ای semigroup روی درخت دودویی

در ابتدا داده ها در گره های برگ ذخیره شده اند و نتایج محاسبات در نودها یا گره های غیر برگ انجام می شود. نتیجه نهایی در نود ریشه قابل دسترسی است.



این محاسبه با اعلام نتیجه در گره ریشه به $\lfloor \log_2 P \rfloor$ زمان نیاز دارد. حال اگر گره ریشه بخواهد نتیجه را به سایر پردازنده ها اعلام کند به $\lfloor \log_2 P \rfloor$ زمان نیاز دارد.

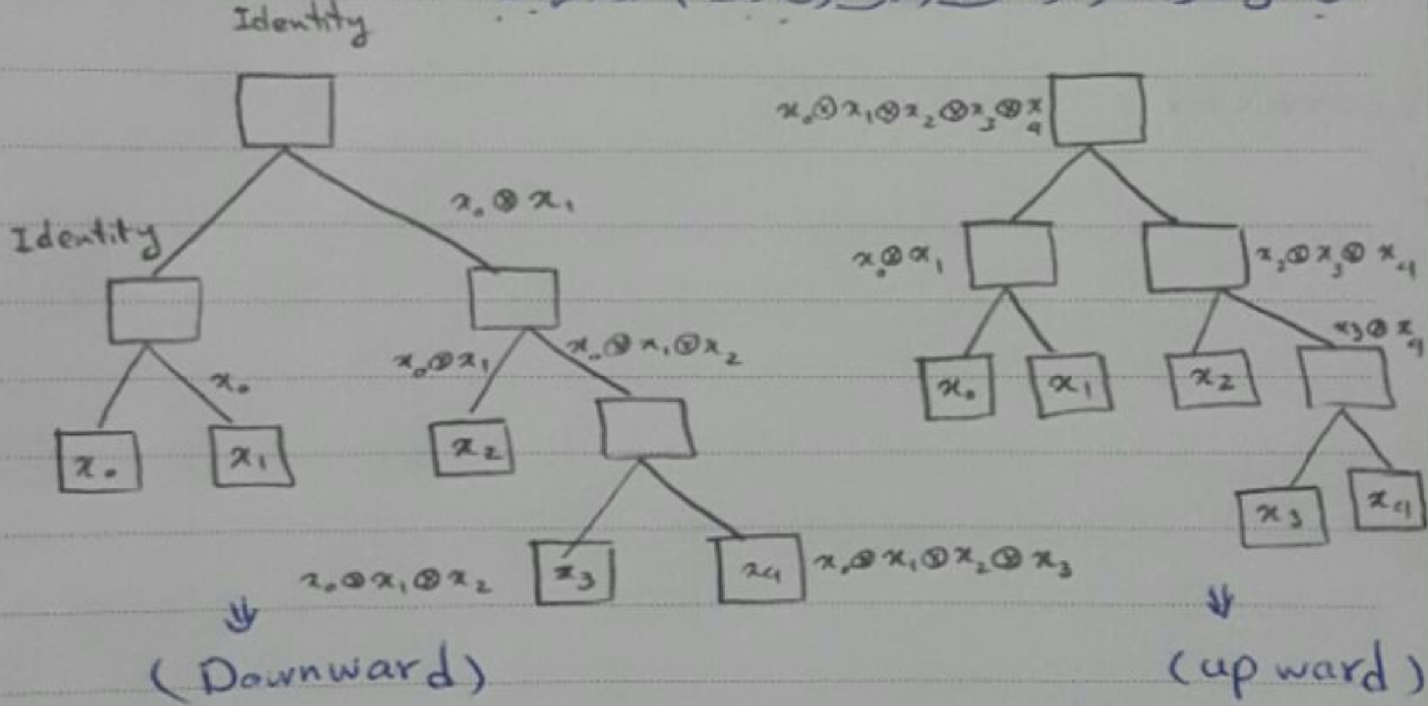
پس در مجموع زمان محاسبه و زمان انتشار (مرحله دوم) به $2\lfloor \log_2 P \rfloor$ نیاز دارد.

Subject:
Year. Month. Date.

• الگوریتم محاسبات موازی (prefix) برای جمع متوالی دودویی :

این الگوریتم در دو فاز انجام می‌شود. در ابتدا از پایین به بالا می‌آیند و محاسبات زنجیره‌ای عمل می‌کنند، در محاسبه‌ی بالا به پایین به این شکل است که هر پردازنده مقداری را که از فرزند چپ خود دریافت می‌کند به خاطر می‌سپارد. هم چنین مقداری را که از گره‌ی پدر دریافت می‌کند به سمت چپ انتقال می‌دهد و فرزند چپ با فرزند سمت راست ترکیب محاسبه می‌شود.

این عمل تا نود آخر (سمت راست ترین برگ) ادامه می‌یابد.



زمان محاسبه دوباره $2\lceil \log_2 p \rceil$

PAPCO

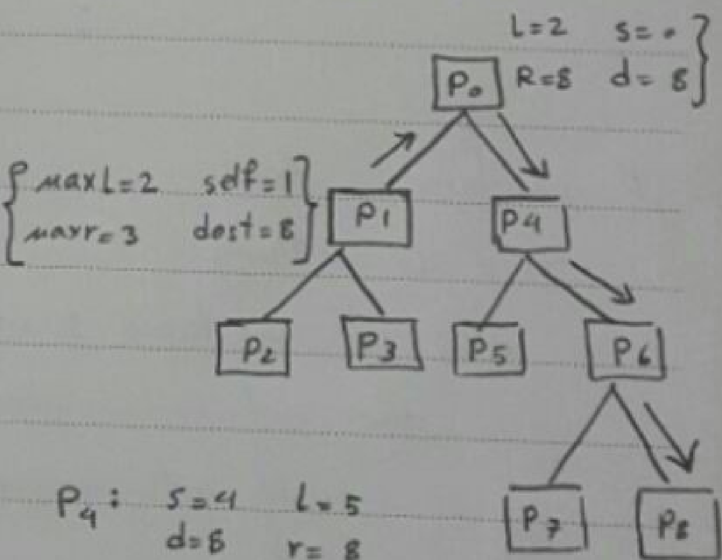
• مسیریابی بسته‌ای packetrouting روی معماری دودویی :

هدف ارسال بسته از پردازنده‌ی i ام به پردازنده‌ی j ام می‌باشد

شرح الگوریتم : اگر گره مقصد همان گره مبدأ است الگوریتم پایان می‌یابد . در غیر اینصورت اگر شماره‌ی گره مقصد > گره مبدأ باشد یا شماره‌ی گره مقصد < بزرگترین شماره در سمت راست باشد حرکت به سمت بالا ادامه می‌یابد .

در غیر اینصورت اگر شماره‌ی گره مقصد ≥ بزرگترین شماره‌ی گره در سمت چپ باشد . حرکت به سمت چپ منتقل می‌شود در غیر اینصورت راست .

```
if dest = self
then remove the packet {done}
else if dest < self or dest > maxr
    then route upward
    else if dest ≤ maxL
        then rout leftward
        else rout rightward
    endif
end if
endif
```



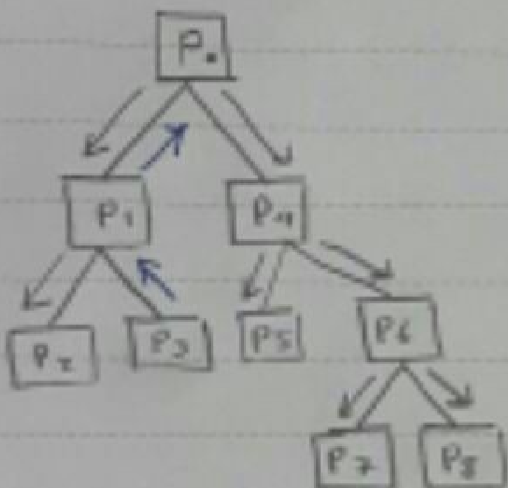
P_4 : s=4 l=5
d=8 r=8

P_6 : s=6
d=8

P_8 : s=8 → پایان
d=8

• الگوریتم Broadcasting (همه‌پخشی) روی درخت عادی دودویی:

پردازنده‌ی i ام می‌خواهد یک پیام را به تمام پردازنده‌ها ارسال کند. ابتدا آن را به ریشه می‌فرستد. سپس ریشه آن را Broadcasting و همه‌پخشی می‌کند.



نکته: زمان حرکت داده از پایین به سمت ریشه و یا از ریشه به سمت کل پردازنده‌ها از log p کمتر نخواهد شد یا حداقل log(log p) باشد.

• الگوریتم مرتب‌سازی (sort) روی درخت دودویی:

```
if you have 2 items
    then do nothing
        else if you have 1 item that com from the left (Right)
            then get the smaller item from the right (left) child
                esle get the smaller item from each child
            end if
            endif
```

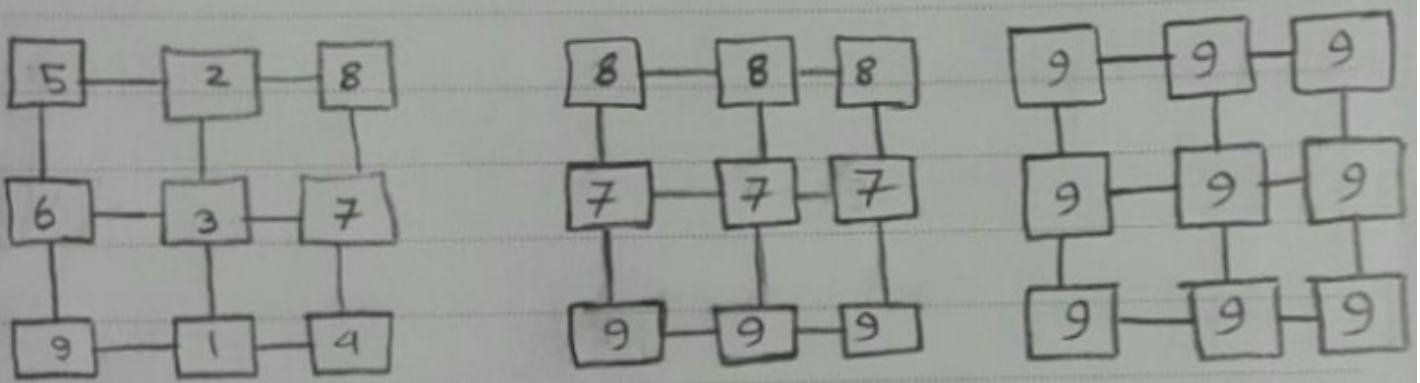
Subject:
Year. Month. Date. ( )

جلسه پنجم پردازش موازی

• الگوریتم semigroup برای یک معماری مش دوبعدی 2D mesh :

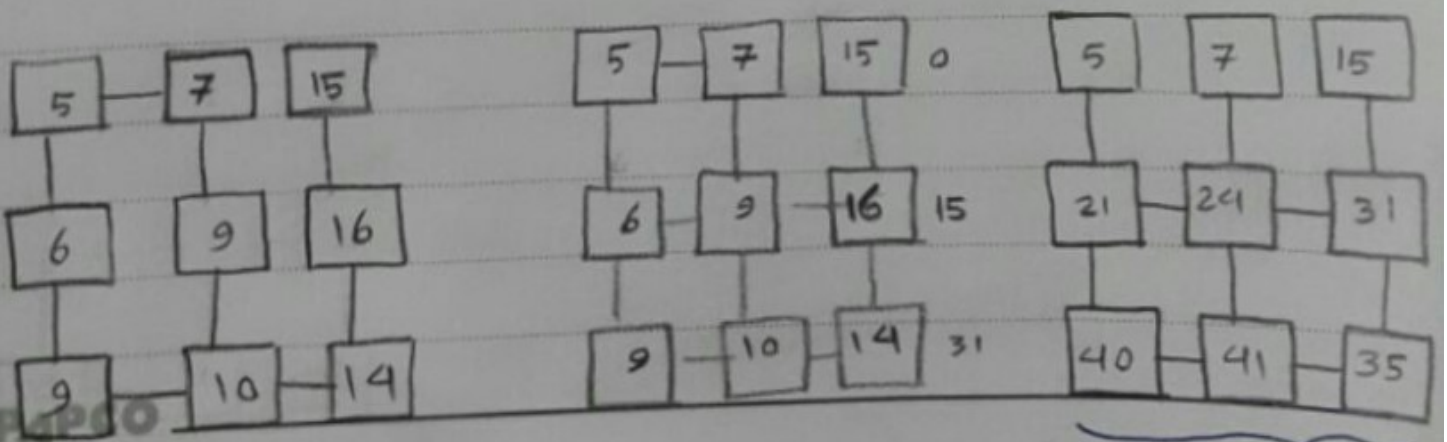
مثلاً در مثال پیدا کردن ماکزیمم اعداد : پردازنده ها در هر سطر به صورت موازی ماکزیمم رقم خود را اعلام می کنند . پس پردازنده ها در هر ستون به صورت موازی max را اعلام می کنند .

در نهایت اعداد ذخیره شده در تمام پردازنده ها نشان دهنده‌ی max اعداد است .



Initial value

• الگوریتم parallel prefix برای یک معماری 2D mesh :



انتشار در سطرها و ترکیب

مثال: حاصلجمع اعداد در $\frac{9}{9}$ پردازنده : در ابتدا در فاز اول پردازنده های هر سطر مجموع

های خود با پردازنده ی آخر هر سطر ذخیره می کند. سپس سمت راست ترین ستون به شکل

prefix کاهش یافته عمل می کند. بدین شکل که نتیجه ی هر پردازنده در آن ستون برابر با

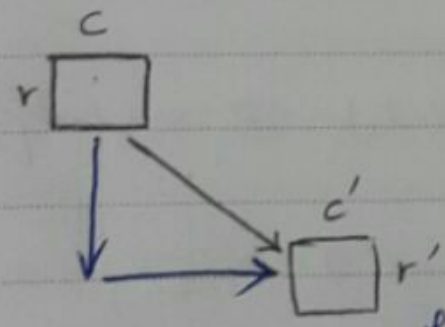
نتیجه ی ذخیره شده در پردازنده های قبلی به اضافه ی داده ی ذخیره شده در خودش می شود.

پس در فاز سوم الگوریتم، نتایج ستون سمت راست در سر تا سر mesh ثبت و می باید

بدین ترتیب در پایان الگوریتم، هر پردازنده نتایج قبلی محاسبات را نیز در خود ذخیره کرده است.

• الگوریتم حرکت packet routing در یک 2D mesh :

به صورت دو الگوریتم بیان می شود.



مثلاً طبق شکل packet(r,c) می خواهد به موقعیت

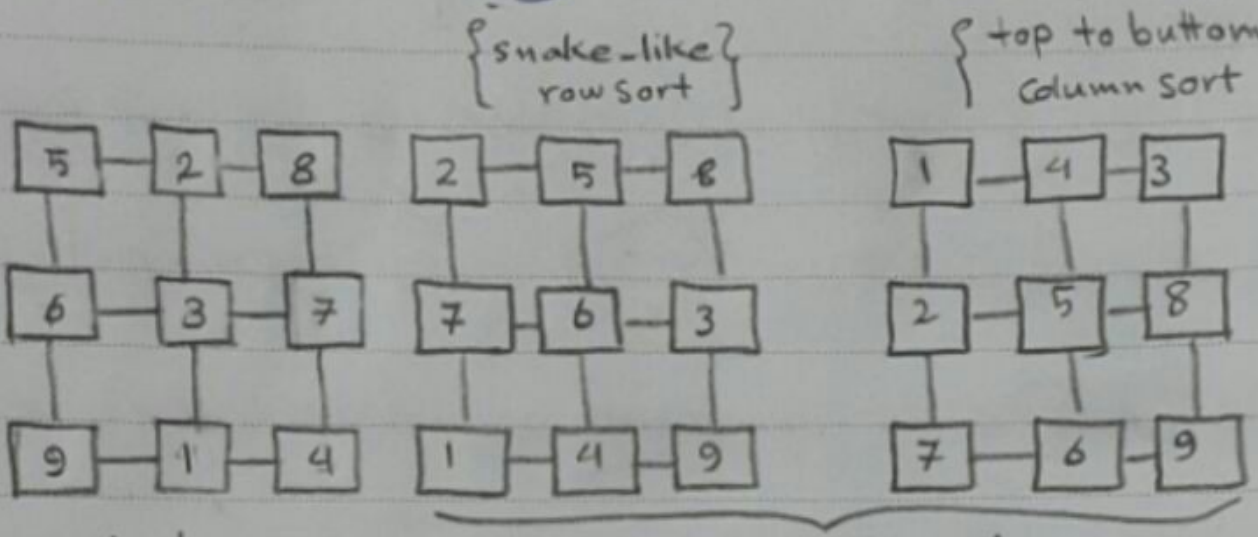
(r',c') برود، در این صورت طبق الگوریتم first_row

می تواند به سطر (r) حرکت کرده پس در طول ستون ها تا ستون (c') پیش برود.

یا به عکس در الگوریتم first_column می تواند به ستون (c) رفته و پس در طول سطر حرکت

Subject:
Year. Month. Date. ( )

• الگوریتم مرتب سازی shearsort برای مش دوبعدی : 2D Mesh



می خواهیم تعدادی عدد را مرتب کنیم . در دو فاز sort میشود .

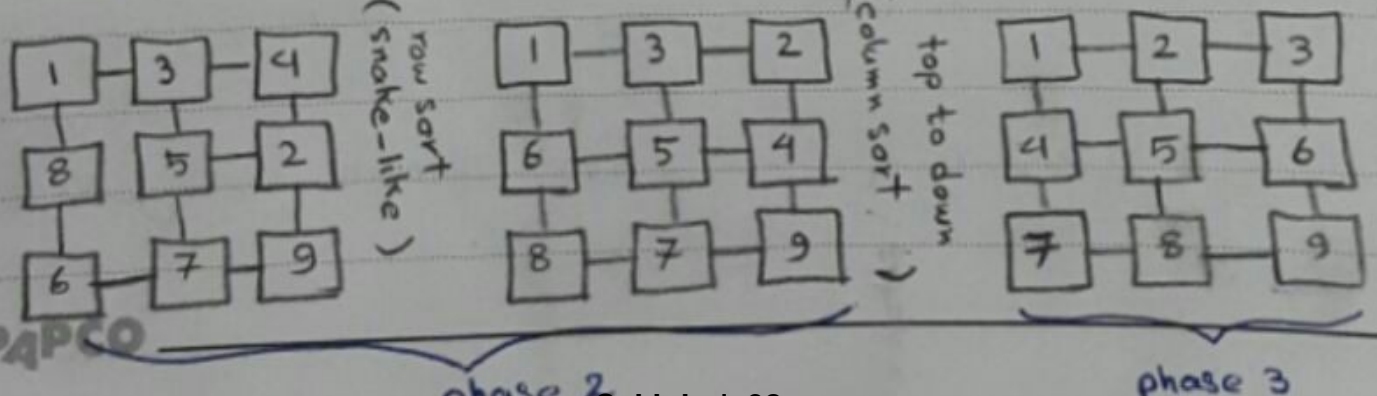
در فاز اول (شکل بالا) سطرهای زوج عمل مرتب سازی را انجام میدهند، سطرهای فرد

به صورت همزمان عمل مرتب سازی را در جهت عکس (راست به چپ) ، انجام میدهند

تمام ستون ها به طور مستقل از بالا به پایین مرتب میشوند

در فاز دوم تمام سطرها از چپ به راست مرتب میشوند ، تعداد تکرارهای فاز 1 الگوریتم

از رابطه ی زیر قابل محاسبه میشود. $1+\lceil \log_2 r \rceil$ ، r : تعداد سطرها



Subject:
Year. Month. Date. ( )

• معماری حافظه‌ی اشتراکی : (مانند رایانه‌های خانه شده) اسلاید 33

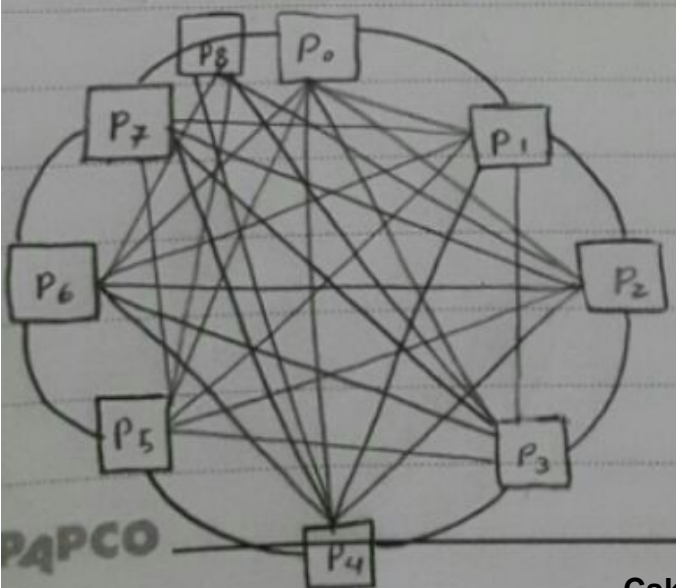
• الگوریتم مرتب‌سازی sheersort بر روی معماری اشتراکی :

هر پردازنده چند عدد در حافظه‌ش ذخیره دارد . هدف مرتب‌سازی اعداد پردازنده به صورت صعودی یا نزولی است .

الگوریتم شامل دو مرحله‌ی ردیف‌ها و ستون‌ها است .

هر پردازنده یک کلید دارد در حافظه‌ش . مثلاً پردازنده‌ی i ام کلید ردیف‌ها به کلید (x_i) است . این عملیات به صورت موازی انجام می‌شود . در حالتی تعداد کلیدها برابر با تعداد پردازنده‌ها باشد . اندیس‌های هر پردازنده برای شناختن ردیف‌ها استفاده می‌شود .

مثال : اگر پردازنده‌ی 3 و 9 هر دو مقدار کلید 23 را داشته باشند ، کلید وابسته به پردازنده‌ی 3 در ردیف کوچکتر قرار می‌گیرد .



Subject:
Year. Month. Date. ( )

روش های اندازه گیری پیچیدگی الگوریتم ها :

• روش دقیق precise ، تعداد اعمال و عملیات های مشخص شده به صورت دقیق و مقدار حافظه ای مصرفی به جدگر جمع شده و برای هر عدد دقیق به عنوان پیچیدگی زمانی ارائه می دهیم .

زمان اجرای تقریبی : (با استفاده از فرمول های تحلیل مجانبی)

$f(n) = O(g(n))$ if $\exists c, n_0$ such that $\forall n > n_0$. $f(n) < c\,g(n)$

$f(n) = \Omega(g(n))$ if $\exists c, n_0$ shuch that $\forall n > n_0$. $f(n) > c\,g(n)$

$f(n) = \Theta(g(n))$ if $\exists c, n_0$ such that $\forall n > n_0$. $cg(n) < f(n) < c'g(n)$

$f(n) = \Theta(g(n))$ iff $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$

Subject :
Year . Month . Date . ( )

جلسه ششم پردازش موازی 95/10/1

• نحوه‌ی دسترسی به حافظه اشتراکی در چند پردازنده‌ها / نحوه‌ی دسترسی پردازنده‌ها به حافظه اشتراکی :

همانطور که قبلاً گفته شد طبق طبقه‌بندی (Flynn) پردازش‌ها در ۴ مدل دسته‌بندی می‌شدند ، MIMD طبقه‌بندی است که بیشتر مدنظر پردازش‌های موازی است ، این طبقه‌بندی مفهوم حافظه‌ی اشتراکی را در خود دارد .

در زیرمدل‌هایی که در این دسته جای می‌گیرند ، هر پردازنده‌ی با روش‌های مختلف ، سه عمل زیر را انجام می‌دهد (همگی از RAM)

✓ عمل دانشی : یعنی دانستن مقدار از آدرس Si در حافظه اشتراکی

✓ انجام محاسبات روی داده‌های قرار گرفته در Register های محلی

✓ ذخیره سازی حاصل در آدرس مقصد در حافظه اشتراکی

نکته : در برخی از حالات ممکن است این سه مرحله به صورت کامل انجام نشود و فقط

برخی از آنها انجام شود. هم چنین آدرس های S_i و d_i در پردازنده ی P_i مستقل از پردازنده های دیگر است.

- انواع زیرمدل های PRAM :

EREW PRAM : exclusive Read exclusive write

وقتی که یک حافظه توسط یک پردازنده اشغال شد، آن خانه مخصوصاً در اختیار آن پردازنده است.

ERCW PRAM : exclusive Read concurrent write

نوشتن
خواندن انحصاری، خواندن هم زمان

وقتی پردازنده‌ای خانه‌ای از حافظه را به منظور خواندن در اختیار گرفت، پردازنده‌های دیگر قادر به دسترسی به آن خانه از حافظه نمی‌باشند اما می‌توانند همزمان خانه‌ای از حافظه را بنویسند (فاقد کارایی)

CREW PRAM : concurrent Read exclusive write

وقتی پردازنده‌ای خانه‌ای از حافظه را به منظور خواندن در اختیار گرفت، پردازنده‌های دیگر نیز

قادر به خواندن از آن خانه حافظه باشند ، اما پردازنده ها نمی‌توانند همزمان خانه‌ای از حافظه را بنویسند .

concurrent Read concurrent write : CRCW PRAM

چند پردازنده می‌توانند همزمان از خانه‌ای از حافظه خوانده یا بنویسند (بسیار قوی)

- انواع CRCW ها :

برای این که مدل CRCW نوشتن همزمان را چگونه انجام می‌دهد ، تعدادی زیر مدل پیشنهاد شده است .

1) CRCW-U (undefined) : مقدار نوشته شده نامشخص است .

2) CRCW-D (Detecting) : تشخیص برخورد به وسیله‌ی تک تک افراد : دقتی چند پردازنده سعی به حافظه‌ی مشترک چند مقدار متفاوت بنویسند ، تک تک افراد آنرا تشخیص می‌دهند و اطلاع می‌دهد که برخورد به وجود آمده است .

Subject:
Year. Month. Date. ( )

3) CRCW_C (common) : اگر مقادیر نوشته شده توسط پردازنده‌ها مختلف یکسان باشند، عمل نوشتن مجاز است (سازگار است)

4) CRCW_R (Random) : مقداری که قرار است در خانه حافظه نوشته شود، به طور تصادفی از پردازنده‌ها انتخاب شود.

5) CRCW_P (priority) : پردازنده با کوچکترین شماره (یا بالعکس) یا اندیس، عمل نوشتن را انجام دهد.

6) CRCW_M (max_MIN) : مقدار نوشته شده برابر با بزرگترین یا کوچکترین مقدار تولید شده توسط چند پردازنده انتخاب می‌شود.

7) CRCW_S (SUM) : مثلاً روی داده‌های تولید شده توسط چند پردازنده که قرار است در خانه‌ای از حافظه نوشته شوند، عمل جمع انجام شده و حاصل جمع در حافظه‌ی مورد نظر نوشته شود.

8) CRCW_A (And)

9) CRCW_X (XOR)

PAPCO