

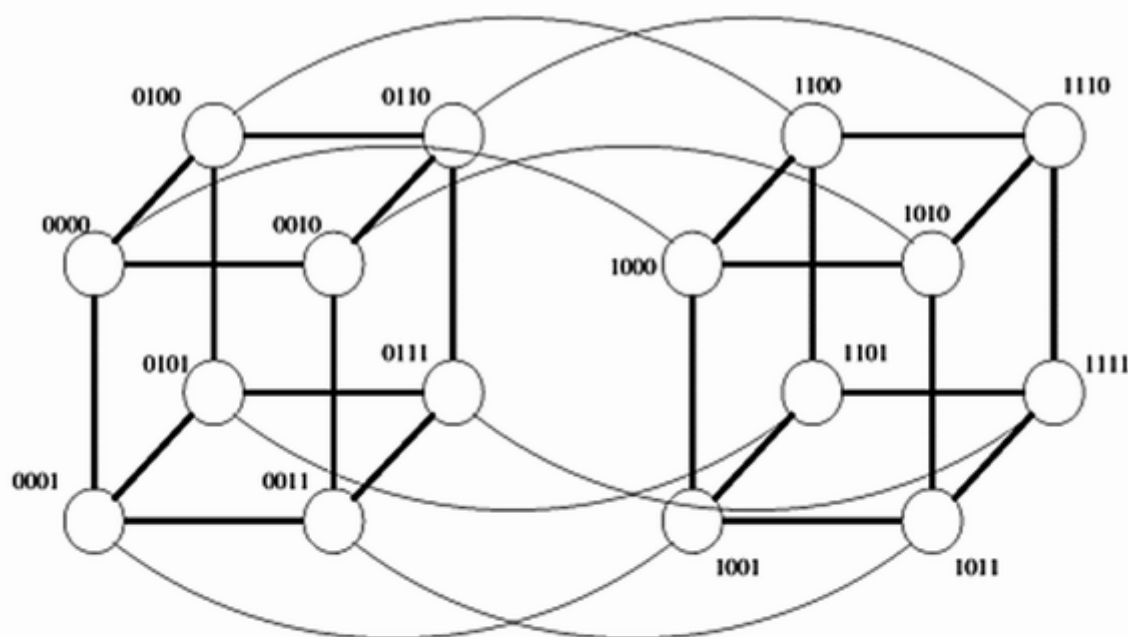
تم تحميل الملف من موقع
البوصلة التقنية
www.boosla.com



المملكة العربية السعودية
وزارة التعليم العالي
جامعة الملك سعود - فرع القصيم
كلية العلوم - قسم الحاسب الآلي

حساب المتوازية

والخوارزميات المتوازية «دراسة»



إشراف

د. عماد فتاش

إعداد

محمد عبدالله الجارالله
بسام عبدالرحمن الخريف
نواف مقبل الحربي
عمر صالح البهدل

المحتويات

٣	 المقدمة
٦	 الفصل الأول: التوازي
٦	 1.1 مفهوم التوازي
٨	 1.2 الحاجة إلى استخدام التوازي
٩	 1.3 فوائد تعدد المعالجات
١٠	 1.4 دراسة المعالجة المتوازية
١١	 1.5 تطبيقات المعالجة المتوازية
١٢	 1.6 تعريف الحاسب المتوازي
١٢	 1.7 التسريع (Speedup)
١٧	 1.8 أشكال معالجة المعطيات على التوازي
١٧	 1.8.1 مستوى البرامج (Programs)
١٨	 1.8.2 مستوى الإجرائية (Procedure)
١٩	 1.8.3 مستوى التعليمات (Instructions)
٢٠	 1.8.4 مستوى التعليمة (Instruction)
٢٢	 1.9 موجز لتاريخ الحاسبات
٢٨	 الفصل الثاني: تصنيف الحاسبات المتوازية
٢٨	 2.1 تصنيف فلاين [Flynn's Classification Scheme]
٢٩	 2.1.1 الحاسبات وحيدة تدفق التعليمات ووحيدة تدفق المعطيات SISD
٣٠	 2.1.2 الحاسبات وحيدة تدفق التعليمات ومتعددة تدفق المعطيات SIMD
٣٤	 2.1.3 الحاسبات متعددة تدفق التعليمات ووحيدة تدفق المعطيات MISD
٣٥	 2.1.4 الحاسبات متعددة تدفق التعليمات ومتعددة تدفق المعطيات MIMD
٣٧	 2.1.4-a الذاكرة المشتركة MIMD Shared Memory
٤١	 2.1.4-b تمرير الرسائل MIMD Message Passing
٤٤	 2.2 شبكات الربط (Interconnection Networks)
٤٥	 2.2.1 الشبكات السكونية
٤٦	 2.2.1.1 الشبكة الخطية والحلقية
٤٦	 2.2.1.2 الشبكة المصفوفية و المصفوفية الحلقية
٤٧	 2.2.1.3 الشبكات الشجرية
٤٨	 2.2.1.4 الشبكات المكعبة
٤٩	 2.2.2 الشبكات الديناميكية
٤٩	 2.2.2.1 شبكة الناقل
٥٠	 2.2.2.2 مصفوفة المبدلات
٥١	 2.2.2.3 الشبكات متعددة الطبقات
٥٣	 الفصل الثالث: مبادئ تصميم الخوارزميات المتوازية
٥٤	 3.1 مفاهيم أساسية
٦٣	 3.2 الإجراءات والمقابلة

٦٦	3.3 تقنيات التقسيم
٦٧	3.3.1 التقسيم العودي (Recursive Decomposition)
٧٠	3.3.2 تقسيم البيانات (Data Decomposition)
٧٧	3.3.3 التقسيم الاستكشافي (Exploratory Decomposition)
٨٢	3.3.4 التقسيم التخميني (Speculative Decomposition)
٨٥	3.3.5 التقسيم المختلط (Hybrid Decompositions)
٨٧	3.4 أمثلة للخوارزميات المتوازية
٨٧	3.4.1 خوارزمية الفرز الفقاعي وتوابعها (Bubble Sort)
٨٨	3.4.1.1 الإبدال الزوجي-الفردى (Odd-Even Transposition)
٩٢	3.4.2 خوارزمية بريم Prim لإيجاد أصغر شجرة هيكلية
٩٢	3.4.2.1 تعاريف ومفاهيم أساسية
٩٥	3.4.2.2 الشجرة الهيكلية الأصغر: (خوارزمية بريم)
١٠١	الفصل الرابع: البرمجة المتوازية
١٠٢	4.1 لغة OCCAM:
١٠٩	4.2 لغة FORTRAN- 90 :
١١٠	4.3 واجهة تمرير الرسائل Message Passing Interface MPI
١١٠	4.3.1 الهيكل العام لبرامج MPI
١١٣	4.3.2 المراسلات (Communicators)
١١٤	4.3.3 الحصول على معلومات عن بيئة التشغيل
١١٦	4.3.4 تراسل البيانات في MPI
١٢٠	4.3.5 برامج تطبيقية باستخدام MPI
١٢١	برنامج لإرسال واستقبال المعطيات
١٢٢	برنامج إرسال المعطيات ضمن حلقة (Ring)
١٢٤	برنامج جمع سلسلة أعداد
١٢٧	برنامج الفرز الزوجي-الفردى
١٣٢	أهم المراجع

المقدمة

...

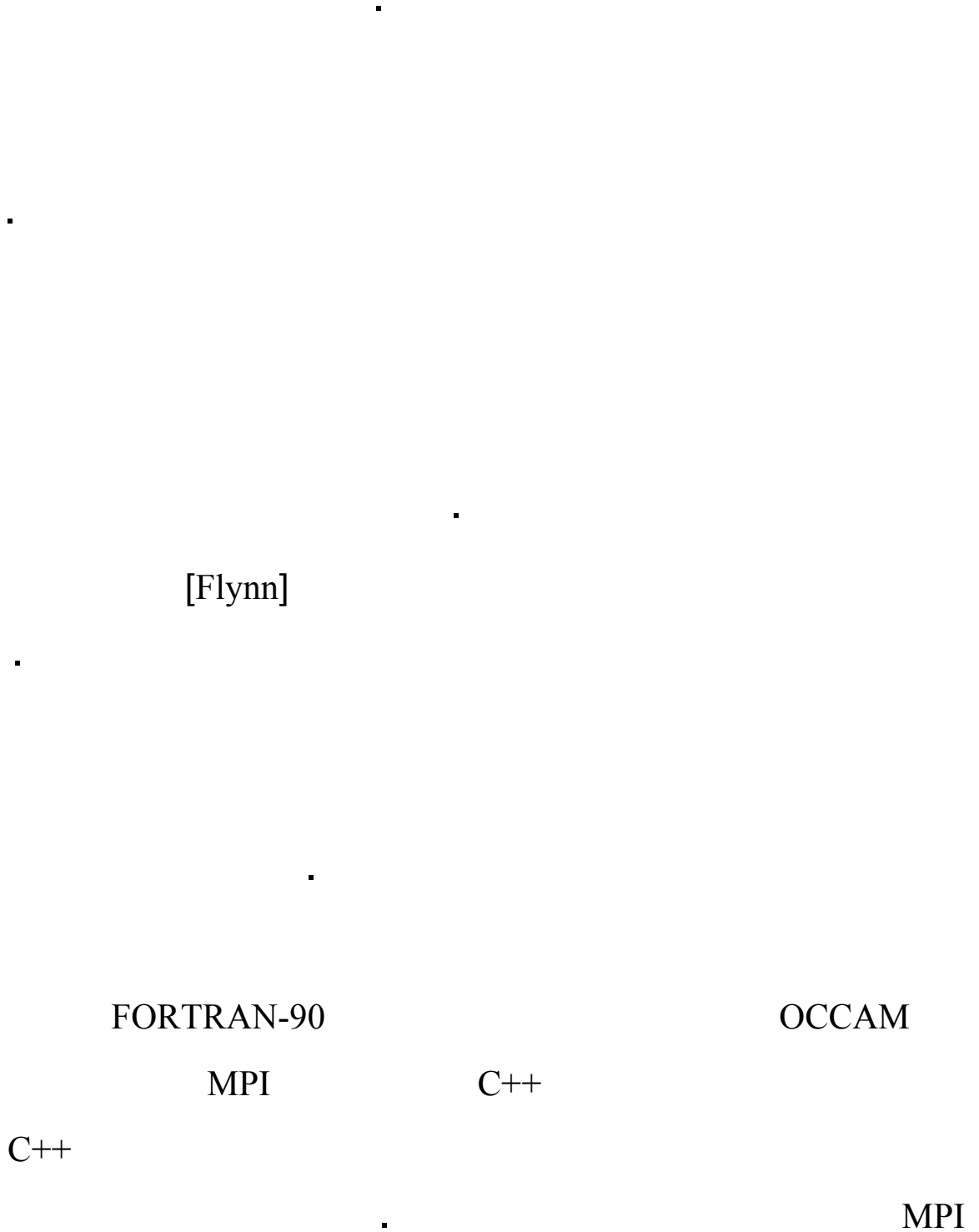
!

■

■

" "

■



هدف المشروع

()

■

■

■

■

⋮

■

■

Parallel_computers@yahoo.com

كافة الحقوق محفوظة

الفصل الأول: التوازي

(adder)

1.1 مفهوم التوازي

"

"

التوازي parallelism: هو مجموعة من الأنشطة التي تحدث في نفس الوقت.

■

)

 $($

11

11

VAX

11

11

11

11

11

11

الحاسبات المتوازية و الخوارزميات المتوازية

التزامن Concurrency: هو القدرة على التشغيل في نفس الوقت.

1.2 الحاجة إلى استخدام التوازي

()

" "

(Supercomputers)

2003

$$\begin{pmatrix} (&) \\ (&) \\ (&) \end{pmatrix}$$

1.3 فوائد تعدد المعالجات

:

-
-
-

-

-

1.4 دراسة المعالجة المتوازية

-

-

(Supercomputers)

-

.(Hardware)

-

"

"

(

)"

"

-

-

المعالجة المتوازية (Parallel Processing): معالجة الحاسب الآلي لعدة برامج في آن واحد (سويًا) وباستعمال عدة وحدات معالجة حسابية ومنطقية.

1.5 تطبيقات المعالجة المتوازية

النمذجة والمحاكاة :

الهندسة :

البحث عن مصادر الطاقة :

.(Mega FLOPS (Floating Point Operation Per Second))

36.5

35 TFLOPS 2003

Bytes

1.6 تعريف الحاسب المتوازي

!

.(datapaths)

الحاسب المتوازي (Parallel computer):

هو حاسب يستخدم عدة معالجات تعمل بشكل متزامن (أي تعمل في نفس الوقت) لحل مسألة أو لأداء وظيفة معينة.

1.7 التسريع (Speedup)

.
 .
 :

·(t_p)

(t_s)

التسريع (S_p):

$$S_p = t_s / t_p$$

· : S_p :

· :t_s

· :t_p

:

n

1/n

:

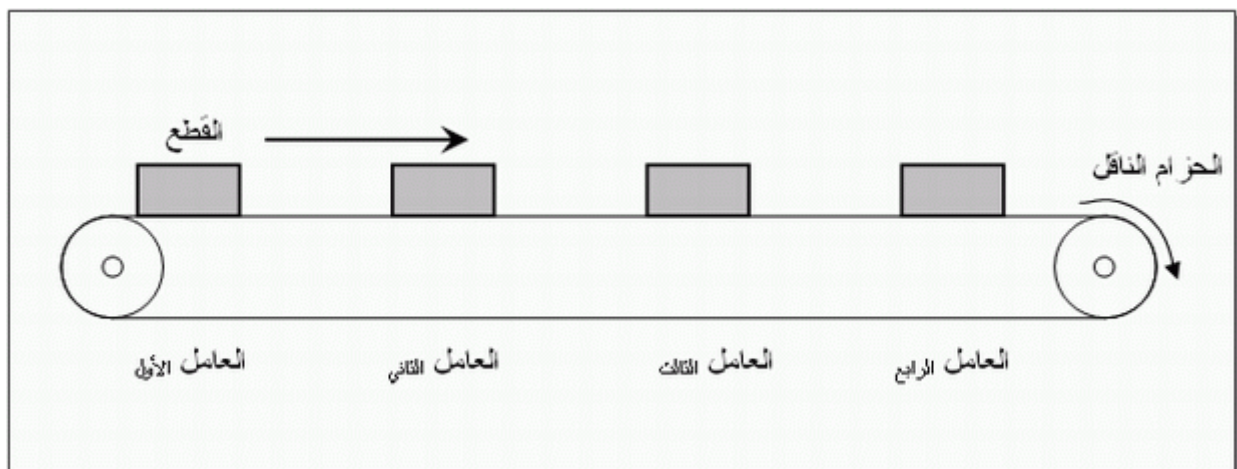
_____ =

·

n

$$S_p = 1 / (1/n) = n$$

$\cdot$ n n
 $\cdot$ n n
 $(\quad)$
 $\cdot$



الشكل (1-1): خط تجميع مكون من أربع محطات عمل

T

...

•

: (T_{w1})

$$T_{W1} = 10 * 4 * T$$

$$= 40T$$

: T :

40T

4T

... 4T

.1T

: (T_L)

$$T_L = 4 * T + (10 - 1) * T$$

:

$$\frac{\text{الزمن لعامل واحد}}{\text{الزمن لخط التجميع}} = \text{التسريع لخط التجميع}$$

:

$$\frac{T_{W1}}{T_L} = \frac{40 T}{13 T} = 3.07$$

.

: : k

$$T_{W1} = T * 4 * k$$

$$T_L = 4 * T + (k - 1) * T :$$

k
: (S_p) ∞

$$S_p = \frac{k * 4 * t}{4 * t + (k - 1) * t} = \frac{4 * k}{3 + k}$$

وبقسمة البسط والمقام على k نستنتج أن :

$$S_p = \frac{4}{\frac{3}{k} + 1}$$

4

K

.

n

:

.(n)

1.8 أشكال معالجة المعطيات على التوازي

:

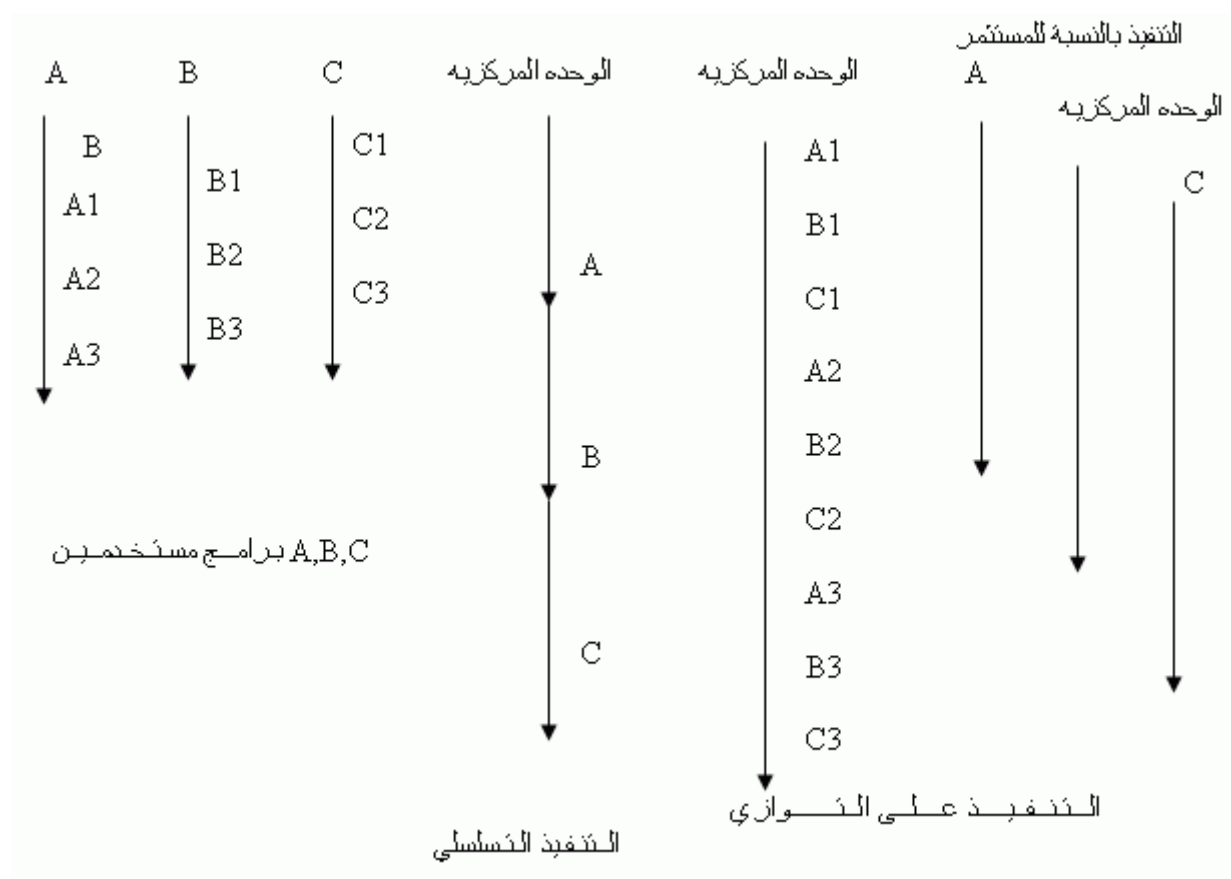
.

1.8.1 مستوى البرامج (Programs)

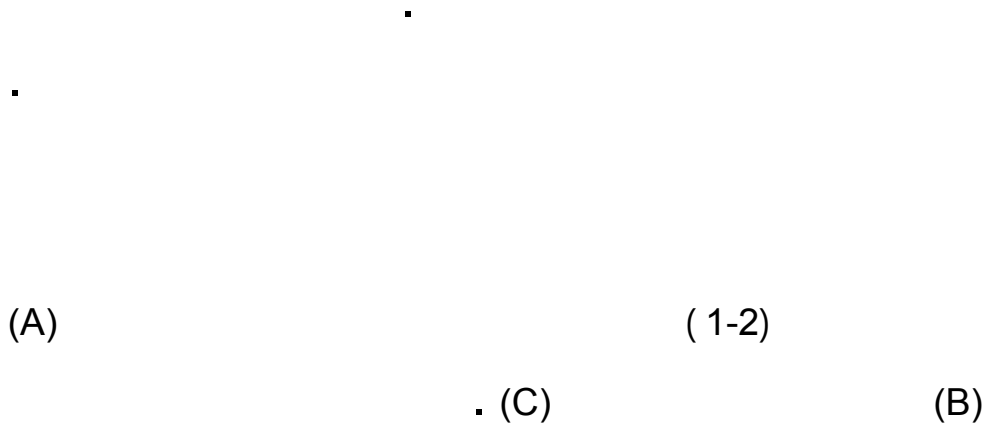
(Temporal Participation)

(Multiprograms)

(Multitreatment)

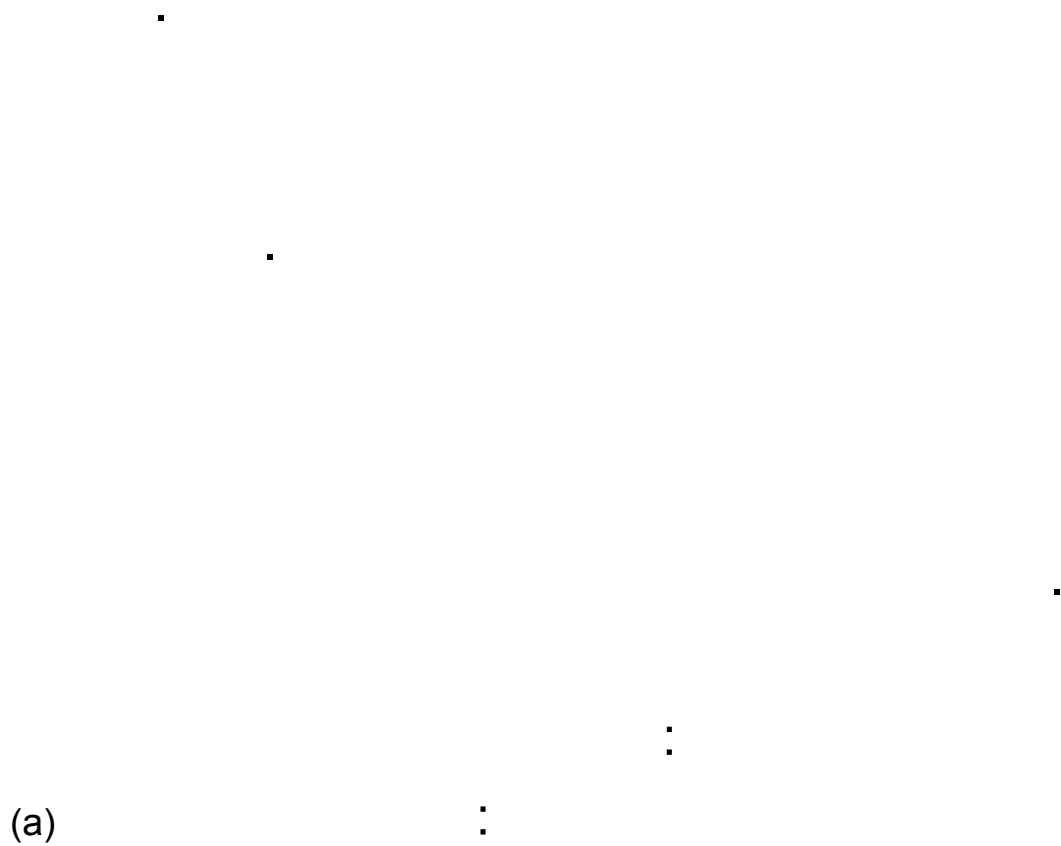


الشكل (1-2): تعددية البرمجيات



· A1,B1,C1,A2,B2,C2,A3,B3,C3:

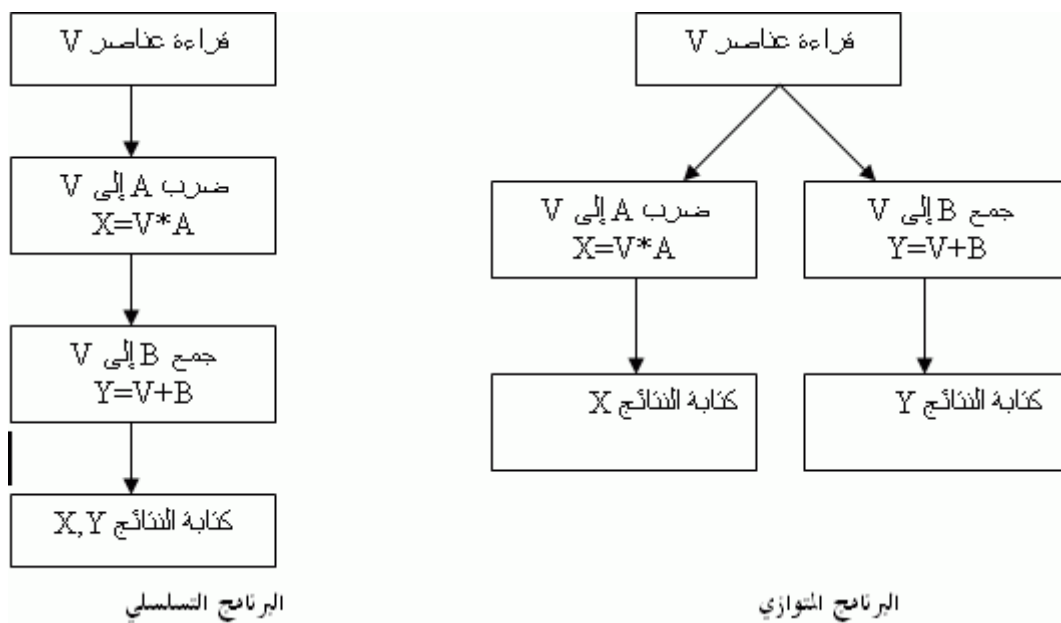
1.8.2 مستوى الإجرائية (Procedure)



(b)

:(1-3)

- .V
- a V
- V b
- .Y X :



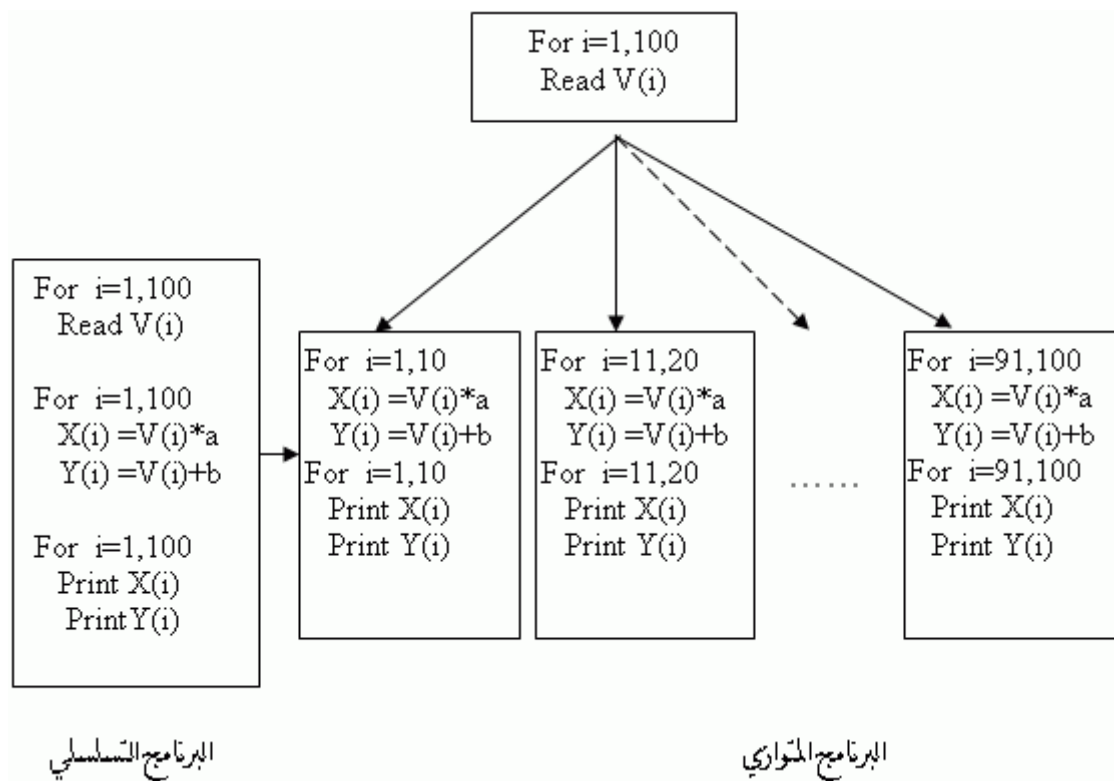
الشكل (1-3) : تحويل برنامج تسلسلي إلى متوازي (مستوى الإجراءات)

1.8.3 مستوى التعليمات (Instructions)

:

. FORTRAN Victories :

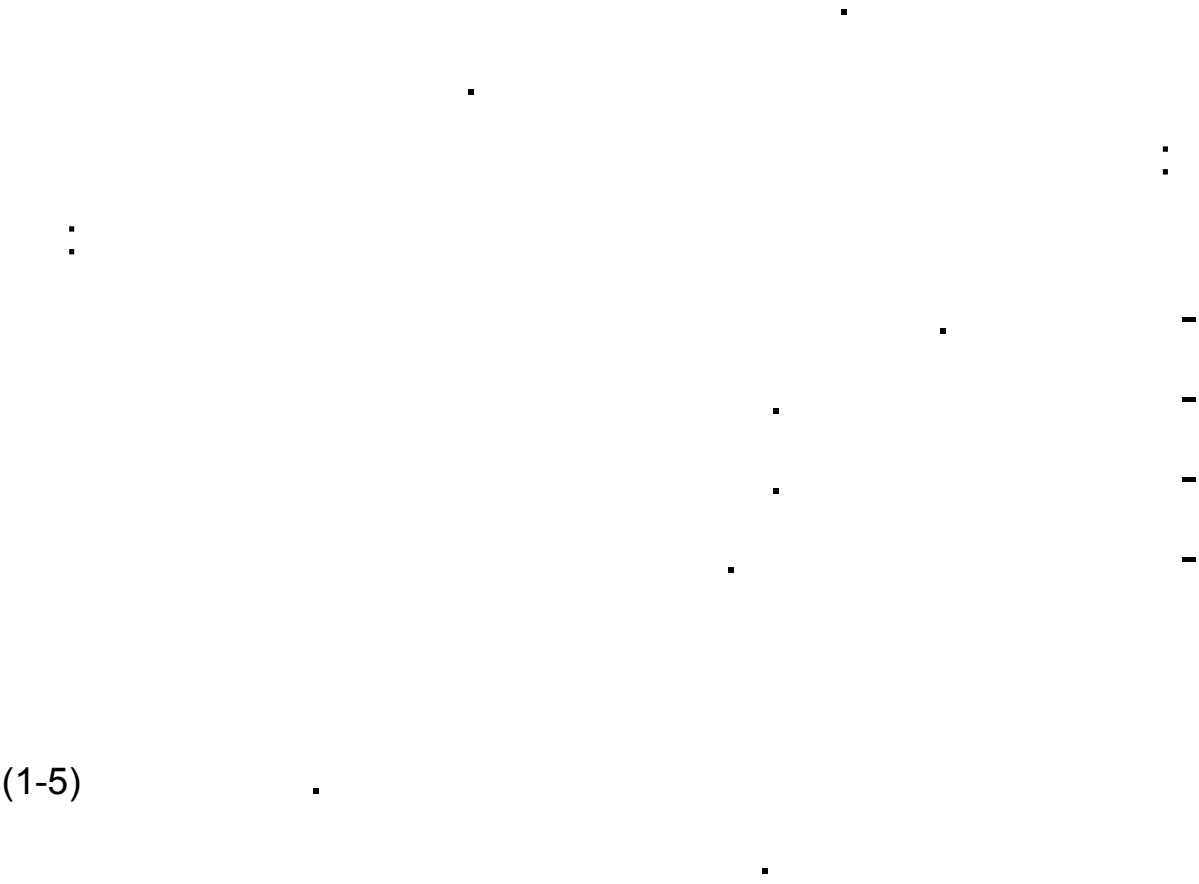
(1-4)



الشكل (1-4) : تحويل برنامج تسلسلي إلى متوازي (مستوى التعليمات)

1.8.4 مستوى التعليمات (Instruction)

(Pipeline)



	ضبط النتيجة	جمع العددين	وضع الرقمين بنفس القوة	مقارنة القوى	
				A.B	المرحلة الأولى
			A.B	C.D	المرحلة الثانية
		A.B	C.D	E.F	المرحلة الثالث
	A.B	C.D	E.F	G.H	المرحلة الرابعة
A.B	C.D	E.F	G.H	I.J	المرحلة الخامسة

الشكل (1-5): العمل الضخمي (Pipeline)

1.9 موجز لتاريخ الحاسبات

(Anacuy)

analog and) " " " "

(digital

•

.()

•

▪

Assembler

:

•

- ENIAC

▪

- EDVAC

- UNIVAC

.1952

الجيل الثاني

1963

1957

:

•

▪

•

▪

•

32000

▪

•

FORTRAN

. (1958) COBOL (1958) ALGOL (1956)

IBM 704

•

▪

▪

▪

IBM 7094

•

(I/O) /

▪

•

.(Stretch

) IBM 7030 LARC

▪

IBM 7030

▪

/

الجيل الثالث

1964

:

) Ics

•

(

.(1972)

Illiac IV

•

▪

•

▪

•

▪

•

•

(microprogramming)

- ## الجيل الرابع

-
-

- ## الحاسبات المتوازية و الخوارزميات المتوازية

أجيال المستقبل

▪

(Artificial Neural Network)

▪

▪

(Hardware)

▪

.(Software)

▪

▪

▪

▪

(Expert System)

▪

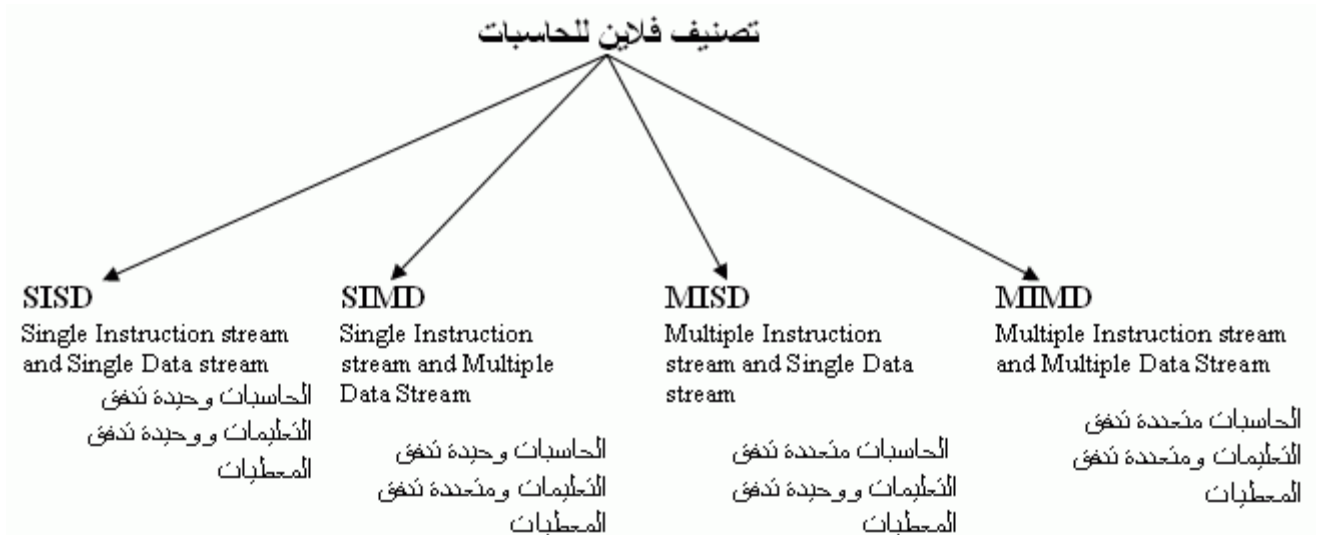
▪

الفصل الثاني: تصنيف الحاسبات المتوازية

1966 [Flynn] " " () :

2.1 تصنيف فلاين [Flynn's Classification Scheme]

()



2.1.1 الحاسبات وحيدة تدفق التعليمات ووحيدة تدفق المعطيات SISD

"Apple Macintosh"

"DEC VAX"

Von

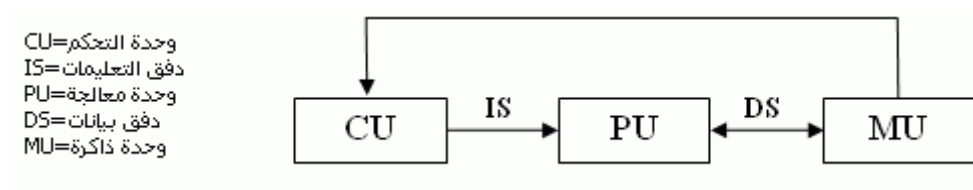
" (SISD)

"

"

"Neumann

(C Fortran)



الشكل (2-2) : SISD

SISD

Cray-1 (Vector)

2.1.2 الحاسبات وحيدة تدفق التعليمات ومتعددة تدفق المعطيات SIMD

(PE)

.(PEs)

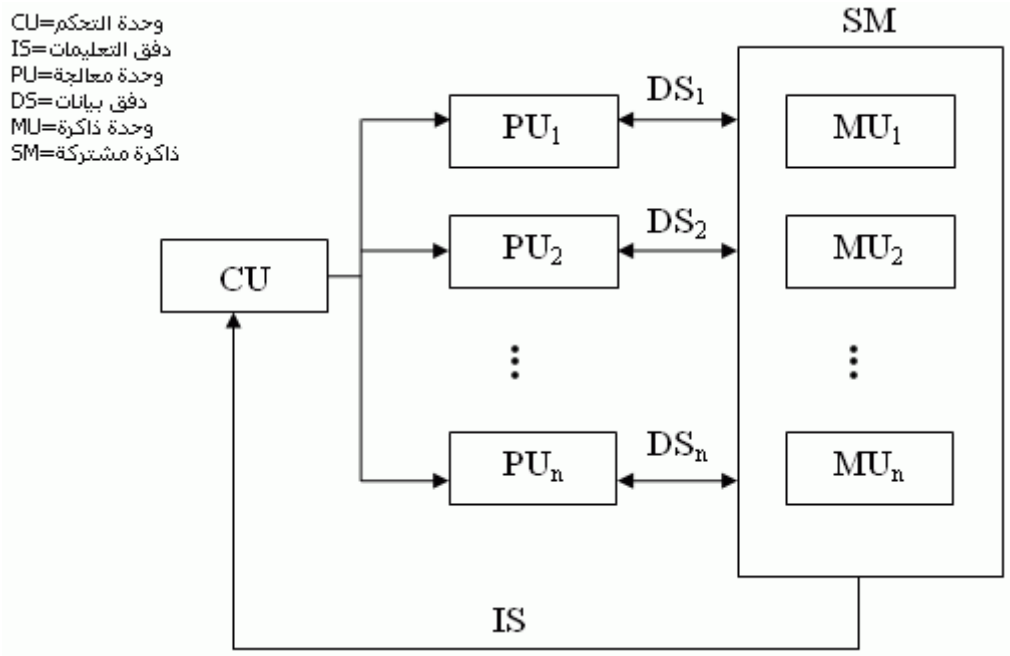
ADD

STORE

SIMD

(2-3)

A(I) B C(I) I=0,...,N A(I)



الشكل (2-3) : SIMD

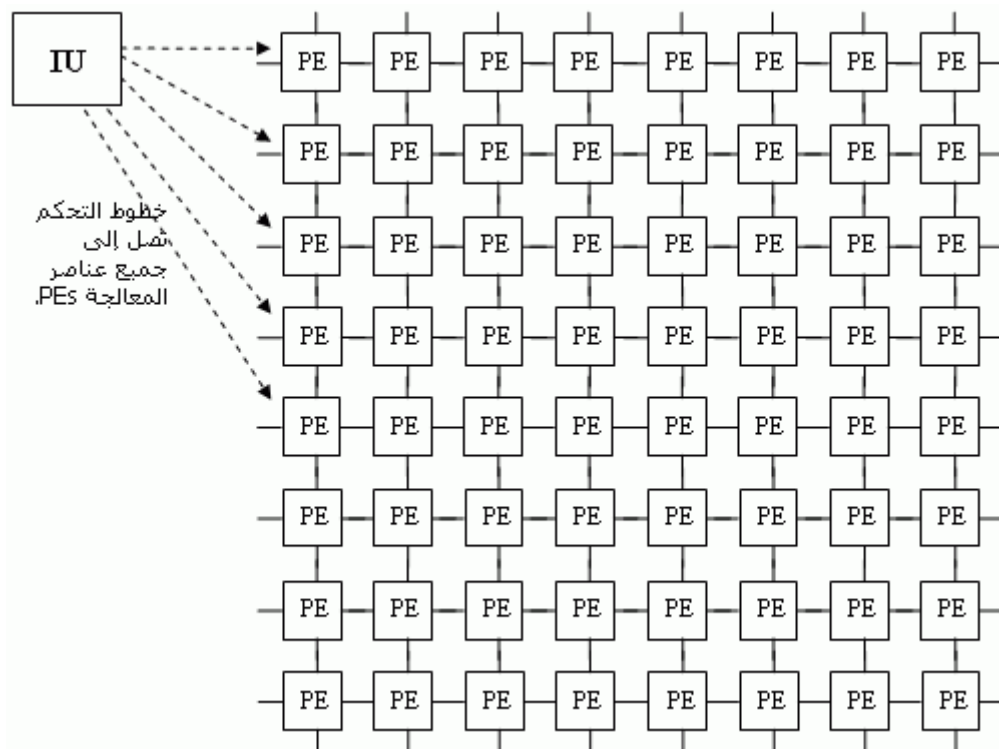
SIMD

(ILLIAC IV)

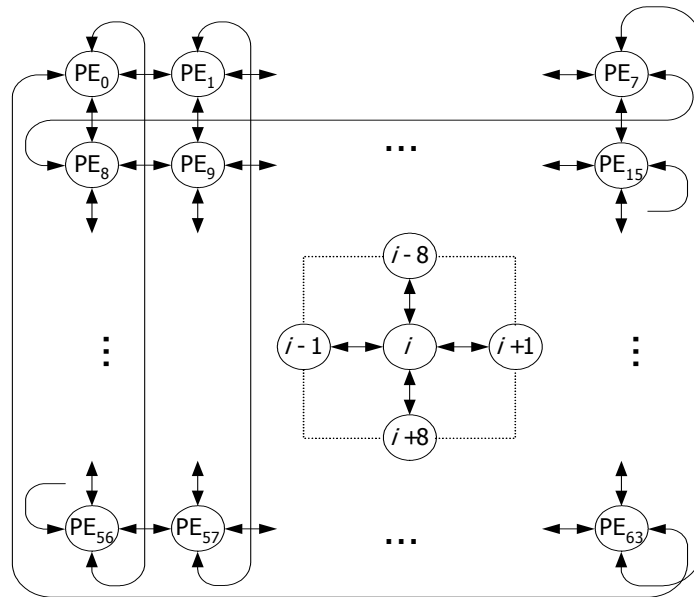
(ILLIAC IV)

(2K words)

. ((2-5)) .



الشكل (2-4): يوضح الجهاز ILLIAC IV مع وحدة تعليمات واحدة (IU) و ٦٤ عنصر معالجة PEs



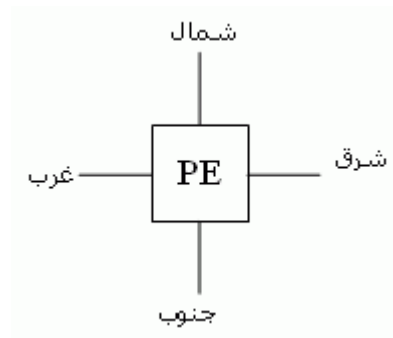
الشكل (2-5): يوضح طريقة التفاف الارتباطات بين المعالجات

(PE)

.(

.NEWS

.



الشكل (2-6): يوضح العنونة باستخدام طريقة شبكة NEWS

(ILLIAC IV)

(ILLIAC IV)

.

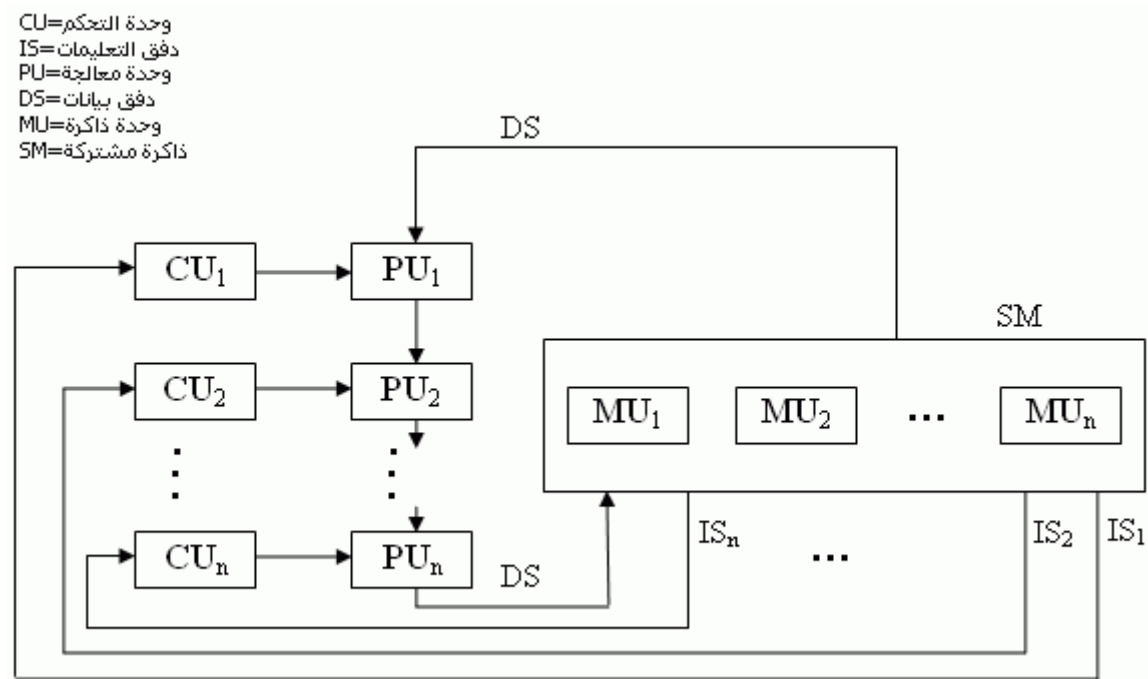
▪
▪
▪
(PE) .

ICL DAP ILLIAC IV : SIMD

Goodyear MPP MasPar MP-2 .

2.1.3 الحاسبات متعددة تدفق التعليمات ووحيدة تدفق المعطيات MISD

▪
▪
MISD
MISD (Flynn) .



الشكل (2-7): MISD

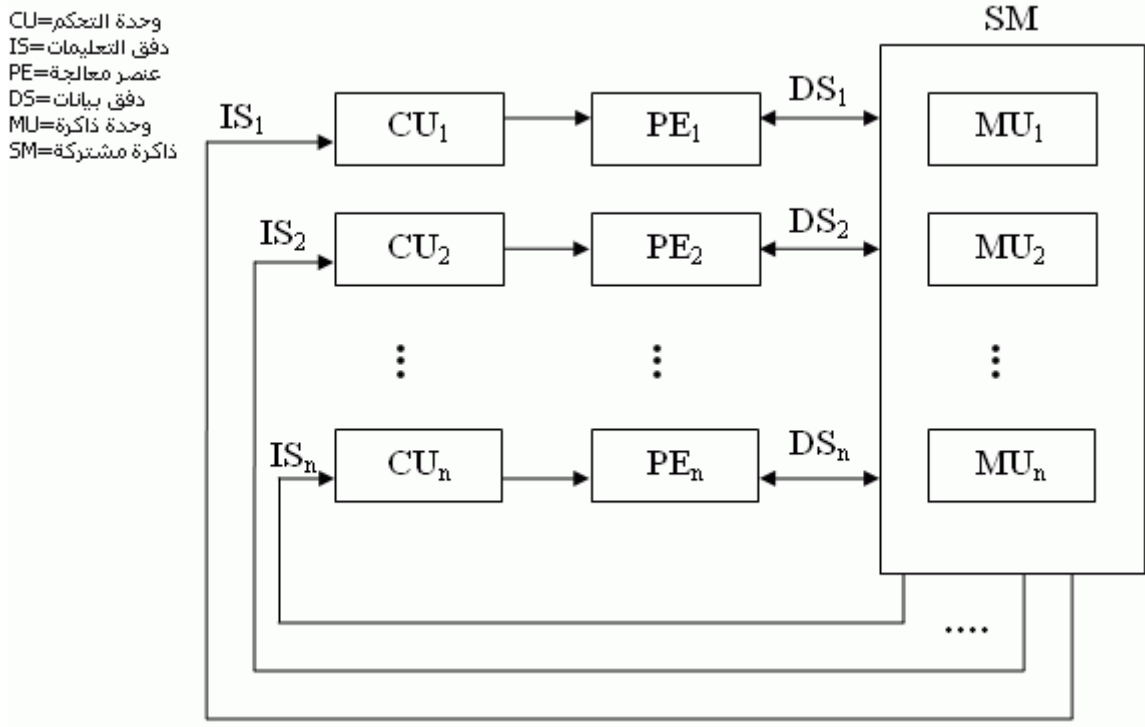
2.1.4 الحاسبات متعددة تدفق التعليمات ومتعددة تدفق المعطيات MIMD

(2-8)

MIMD

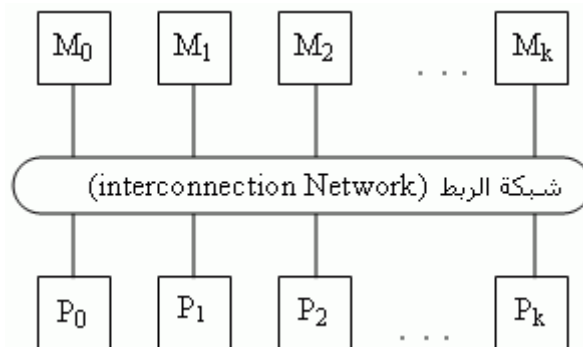
(a) الذاكرة المشتركة (Shared memory).

(b) تمرير الرسائل (message passing).



الشكل (2-8): MIMD

2.1.4-a MIMD Shared Memory الذاكرة المشتركة



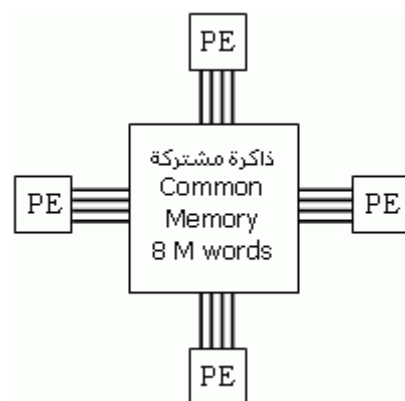
الشكل (2-9): نموذج الذاكرة المشتركة (MIMD Shared Memory)

.(Mesh)

Cray X-MP

النموذج الأول:

(2-10) .



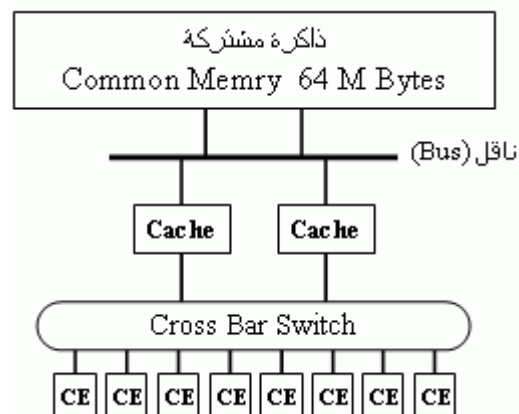
الشكل (2-10): يوضح الجهاز Cray X-MP/48 وكل معالج (PE) له أربع منافذ للذاكرة المشتركة

النموذج الثاني: The Alliant FX/8 minisupercomputer

(CEs)

Crossbar Switch

(cache)



الشكل (2-11): يوضح الجهاز Alliant FX/8 بنمائية معالجات (CEs) تشترك في الذاكرة.

النموذج الثالث: The Bolt, Beranek and Newman (BBN) Butterfly

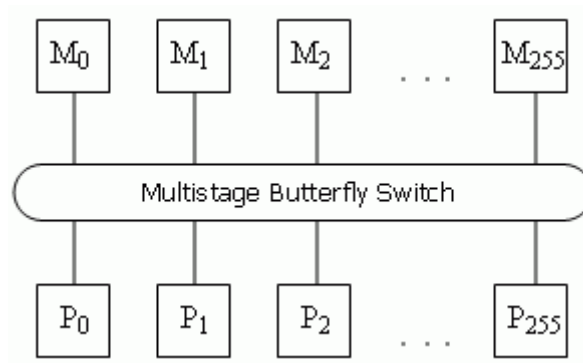
1983

MC68000

256 2

4

(butterfly network).



الشكل (2-12): الجهاز BBN Butterfly يمكن فيه لأي عنصر معالجة P أن يتحول إلى أي وحدة ذاكرة M

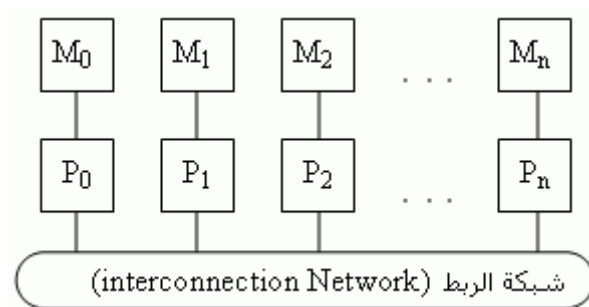
MIMD

(hot spot)

compiler

2.1.4-b تمرير الرسائل MIMD Message Passing

(interconnection Network). (2-13)



الشكل (2-13): يوضح نموذج تمرير الرسائل

.

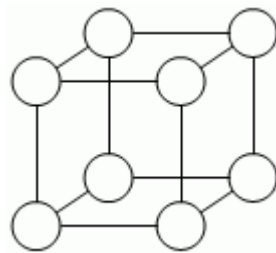
n " "

"

... .

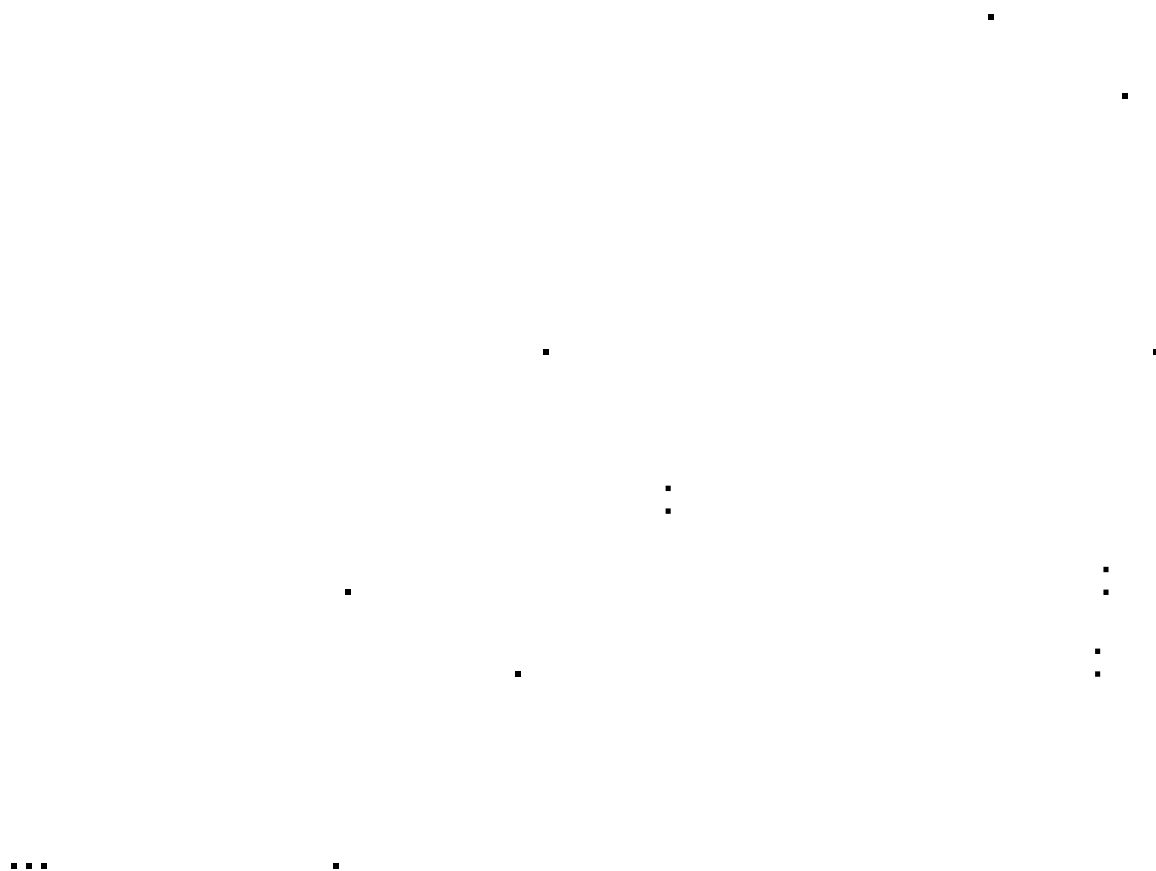
(n-dimensional) "

:(2-14)



الشكل (2-14): مكعب ثلاثي الأبعاد، المعالجات تتوضع على زواياه

¹ في الصفحات من ٤٤ وحتى ٥٢ شرح لشبكات الربط.



CRAY X-MP

(2-10)

128 Intel iPSC ...
nCUBE .(7 $2^7 = 128$)
.(13 $2^{13} = 8192$) 8192

SISD,)
... (SIMD, MIMD
dataflow . ICL DAP
reduction .

2.2 شبكات الربط (Interconnection Networks)

▪
(switches)

▪
(Routers).

▪

▪

▪

▪

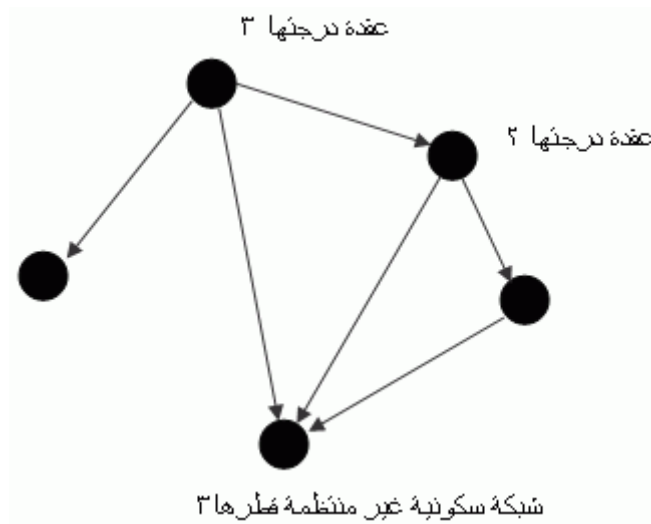
▪

▪

▪

▪

2.2.1 الشبكات السكونية



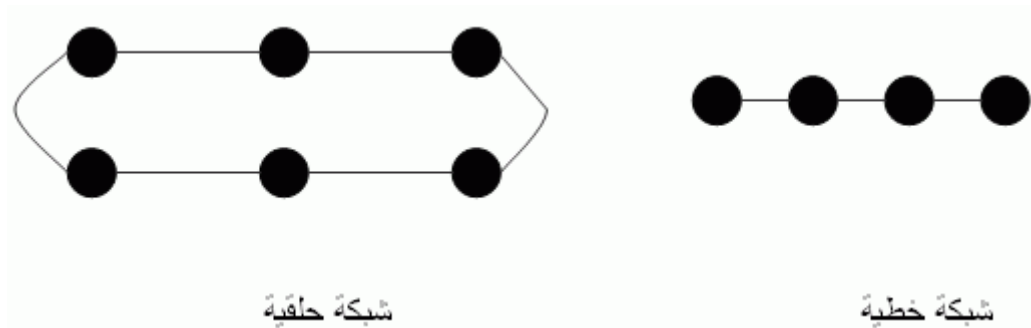
الشكل (2-15) : تمثل الشبكة السكونية بمخطط بياني

Regular

2.2.1.1 الشبكة الخطية والحلقية

2

.M bits/s



الشكل (2-16)

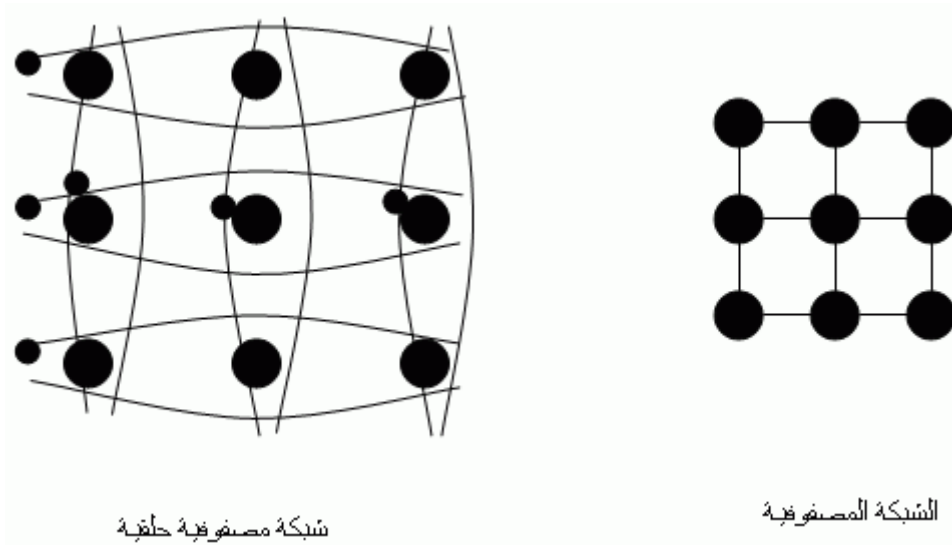
i860

Paragon

2.2.1.2 الشبكة المصفوفية و المصفوفية الحلقية

. (2 or 3)

.(2-17)



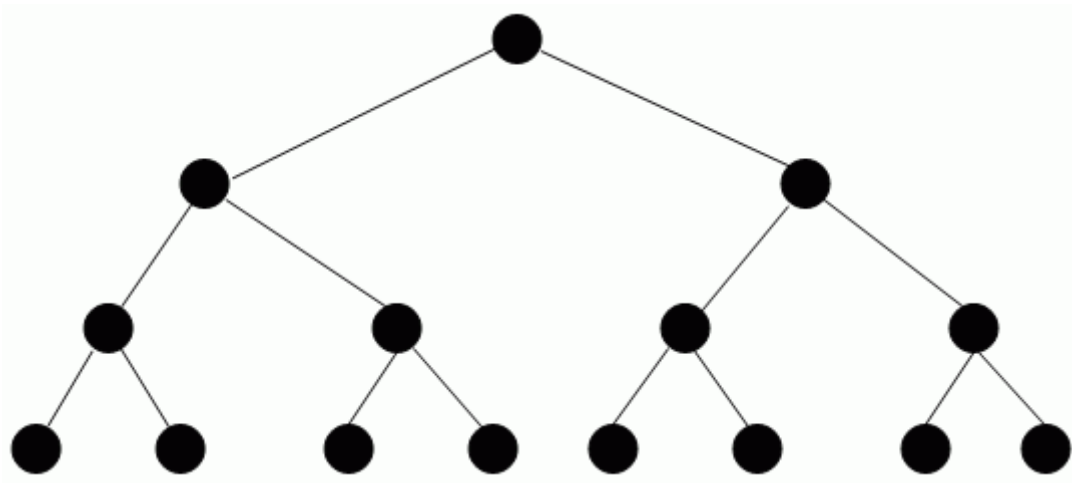
الشكل (2-17): الشبكة المصفوفية و المصفوفية الحلقية

2.2.1.3 الشبكات الشجرية

· (2-18)

(
:

· CM-5



الشكل (2-18) : الشبكة الشجرية

2.2.1.4 الشبكات المكعبة

·

n

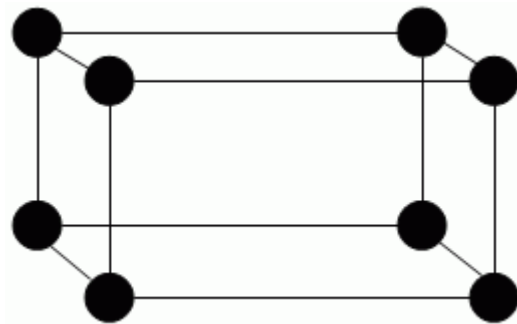
2^n n

·

· (2-19)

$n-1$

(n)



الشكل (2-19) : شبكة مكعبية من الدرجة ٢

2.2.2 الشبكات الديناميكية

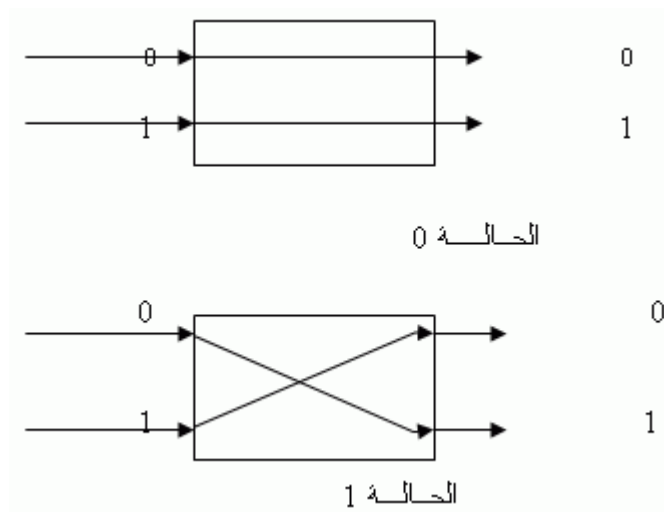
(Router) (Switchs)

2.2.2.1 شبكة الناقل

(First In First Out)FIFO

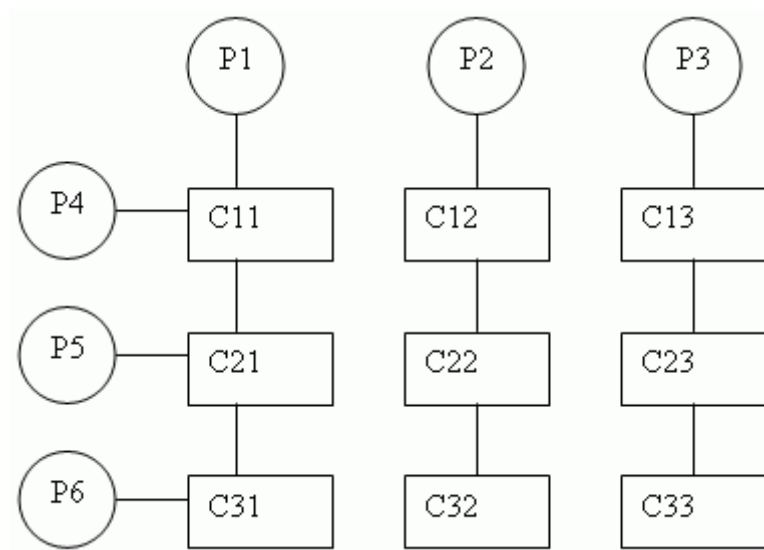
2.2.2.2 مصفوفة المبدلات

(2-20) :



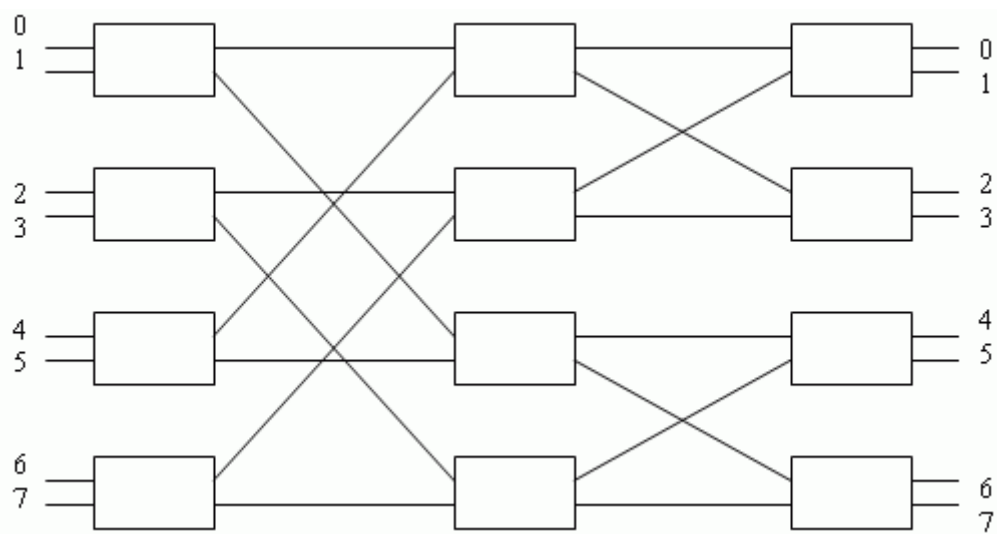
الشكل (2-20)

(0)	(0)	(0)	:(0)
(1)	(0)	(0)	.(1)
(0)	(0)	(1)	:(1)
(1)	(1)	(1)	.(1)



الشكل (2-21) : مصفوفة المبدلات

2.2.2.3 الشبكات متعددة الطبقات



الشكل (2-22) : شبكة متعدد الطبقات

n

$n \cdot \log_2(n)$

$\cdot \left(\frac{n}{2} \right)$

.(2-22)

$\log_2(n)$

الفصل الثالث: مبادئ تصميم الخوارزميات المتوازية

)

(

■

□ □

)

$$(\quad)$$
 $($

■



●

●

●

●

●

3.1 مفاهيم أساسية

• تصميم الخوارزميات المتوازية:

- :

-

:

.

• التقسيم (Decomposition):

• المهام (Tasks):

(Concurrency)

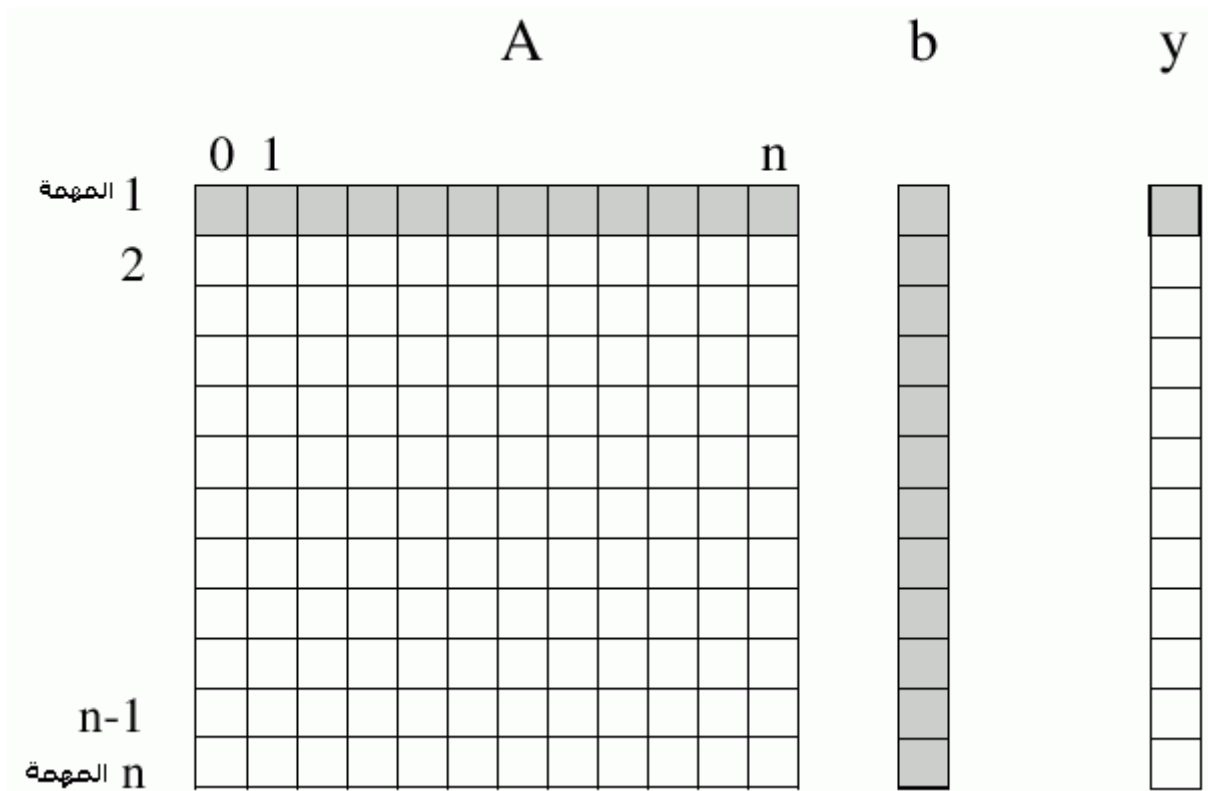
• مخططات التبعية (Dependency Graph):

/

مثال (3-1): ضرب مصفوفة بشعاع

$$\begin{array}{c}
 b \quad n \times n \quad A \\
 i \quad y[i] \quad .y \\
 y[i] = \sum_{j=1}^n A[i, j].b[j] : \quad .b \quad A \\
 y[i] \quad (3-1)
 \end{array}$$

(3-4)

. $n/4$ 

الشكل (3-1): مسألة ضرب مصفوفة بشعاع مقسمة إلى n مهمة، حيث n هي عدد أسطر المصفوفة. الجزء الذي تتعامل معه (مدخلات ومخرجات) للمهمة 1 موضح باللون الغامق.

(3-1)

.

.

)

.

(

)

.(

مثال (3-2) إجرائية الاستعلام من قواعد البيانات

(3-1)

ID

.. color year

ID#	Model	Year	Color	Price
4523	Civic	2003	Blue	55,000
3476	Corolla	1999	White	45,000
7623	Camry	2003	Green	59,500
9834	Prius	2001	Green	48,000
6734	Civic	2001	White	47,000
5342	Altima	2001	Green	49,000
3845	Maxima	2001	Blue	52,000
8354	Accord	2000	Green	48,000
4395	Civic	2001	Red	47,000
7352	Civic	2002	Red	48,000

الجدول (3-1): قاعدة بيانات لتخزين معلومات عن السيارات.

:

MODEL="Civic" AND YEAR="2001" AND (COLOR="Green" OR
COLOR="White")

Civic

:

.

(Relational Database)

□

□

.Civic



■

11

11

"Civic

||

Civic

11

11

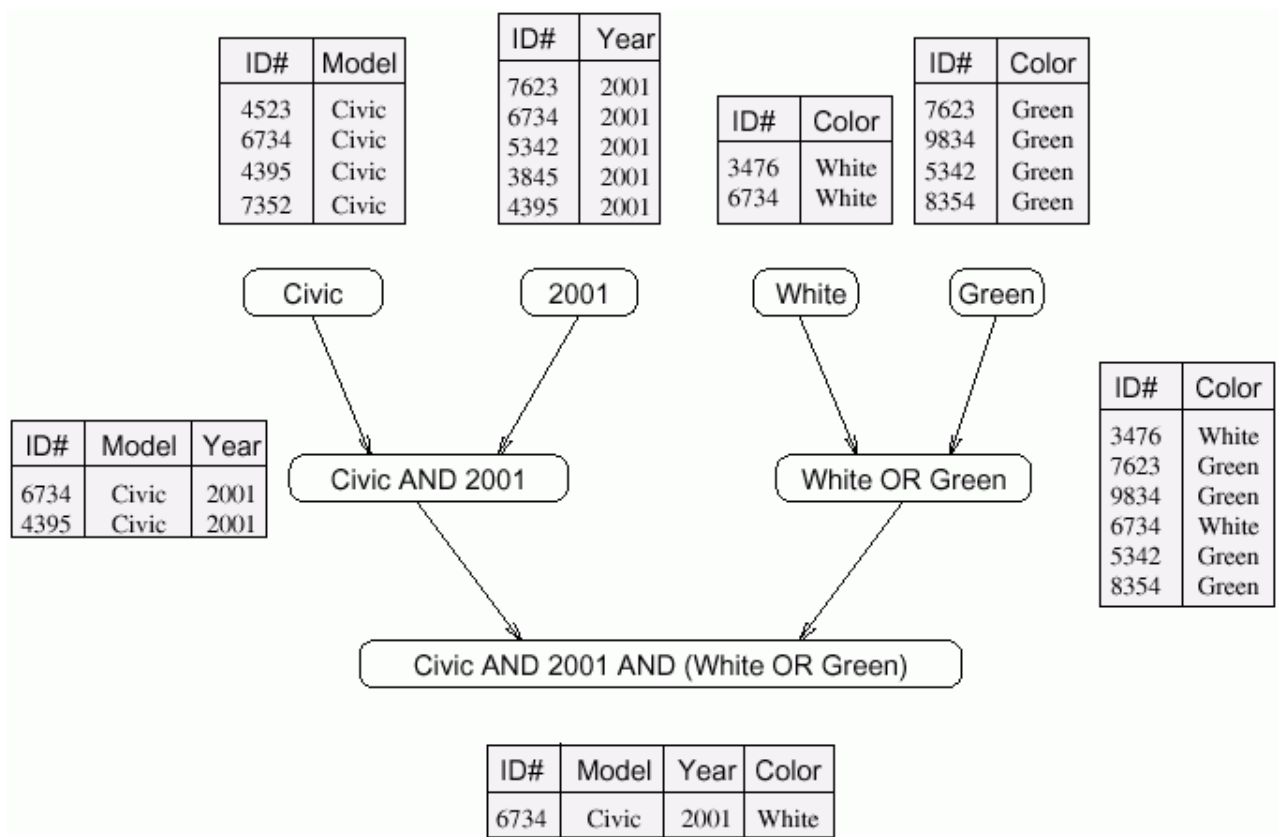
11

11

■

Civic 2001

.(3-2)



الشكل (3-2): الجداول المختلفة والعلاقة بينها في عملية الاستعلام.

()

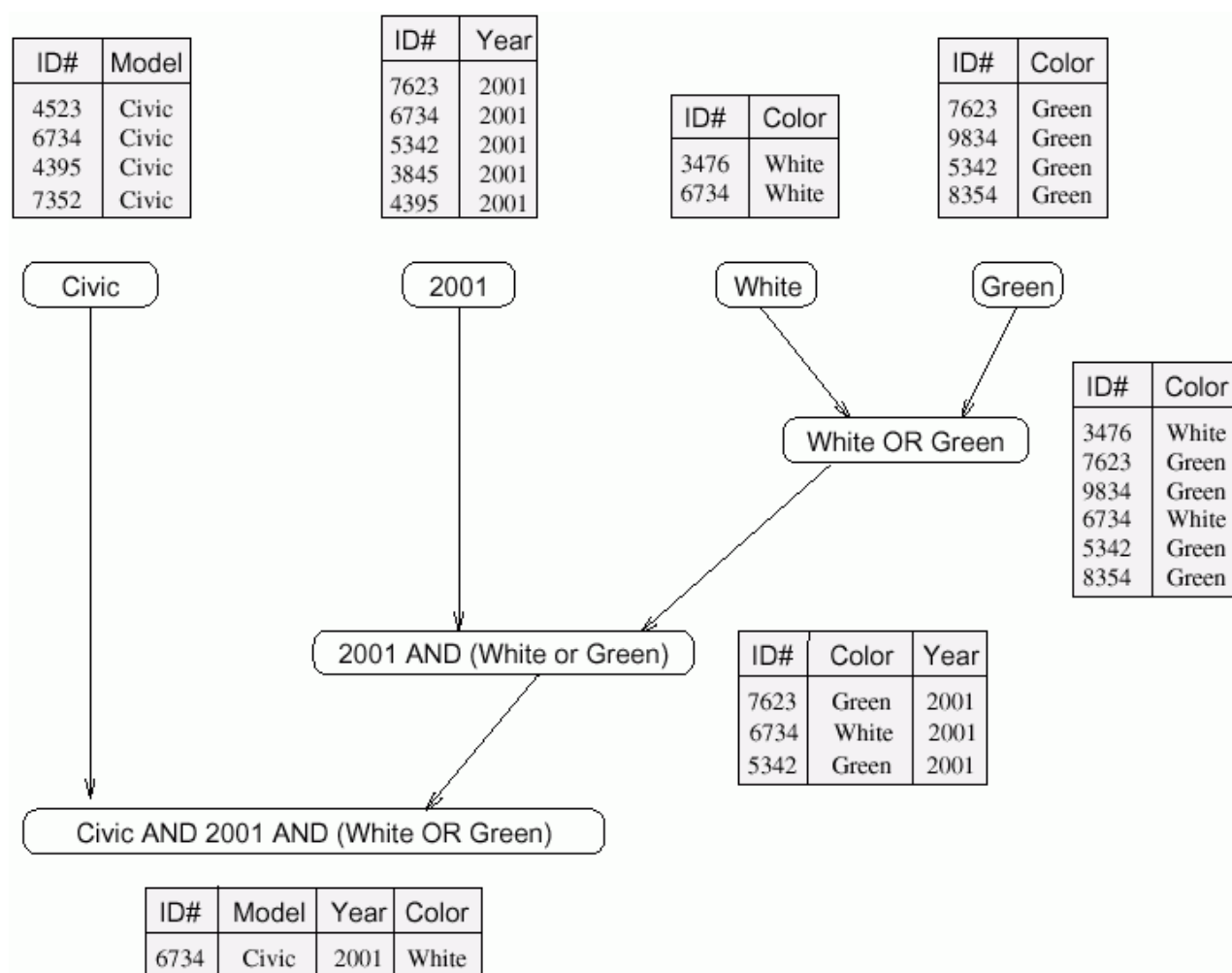
Civic 2001

" "Civic "

OR AND :

(3-2)

: " " :
 " "
 : "Civic" :
 : (3-3)



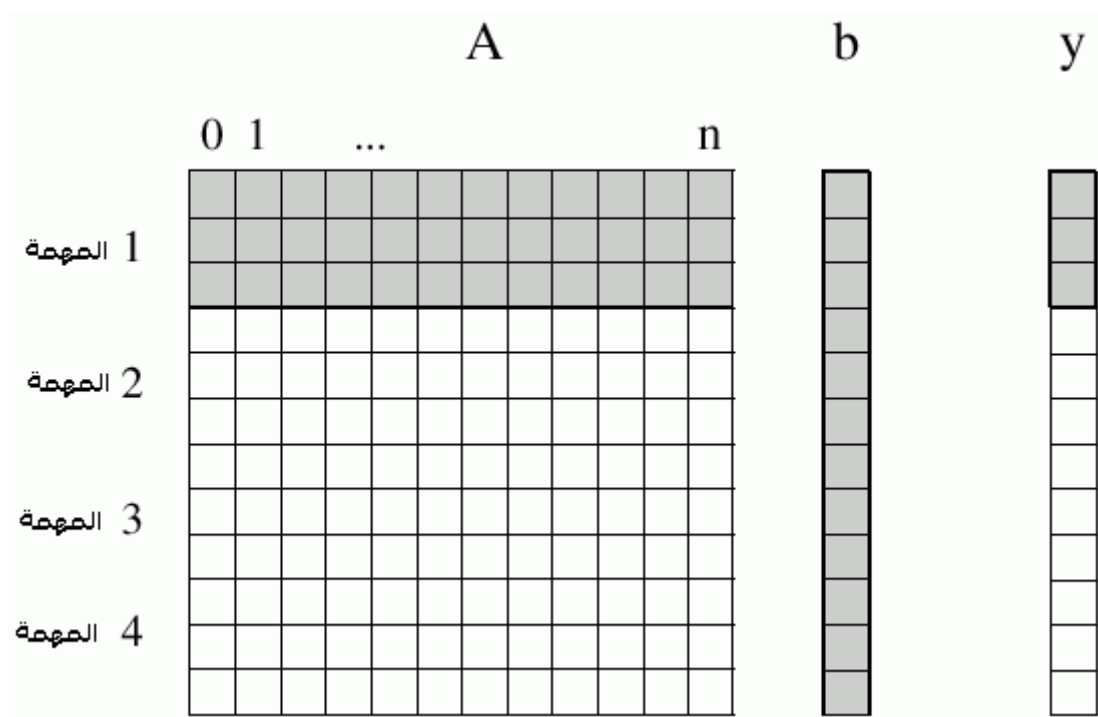
الشكل (3-3): مخطط التبعية لعملية الاستعلام.

• الحبوبية (Granularity):

(3-1)

(3-4)

$n/4$



الشكل (3-4): مسألة ضرب مصفوفة بشعاع مقسمة إلى أربعة مهام. الجزء الذي تتعامل معه (مدخلات ومخرجات) للمهمة 1 موضح باللون الغامق.

"

 .

 .4 (3-3) (3-2)

)

 (

 .

 "

 "

 .

 .()

 (3-1)

 (3-4)

 .

 (3-5)

 .

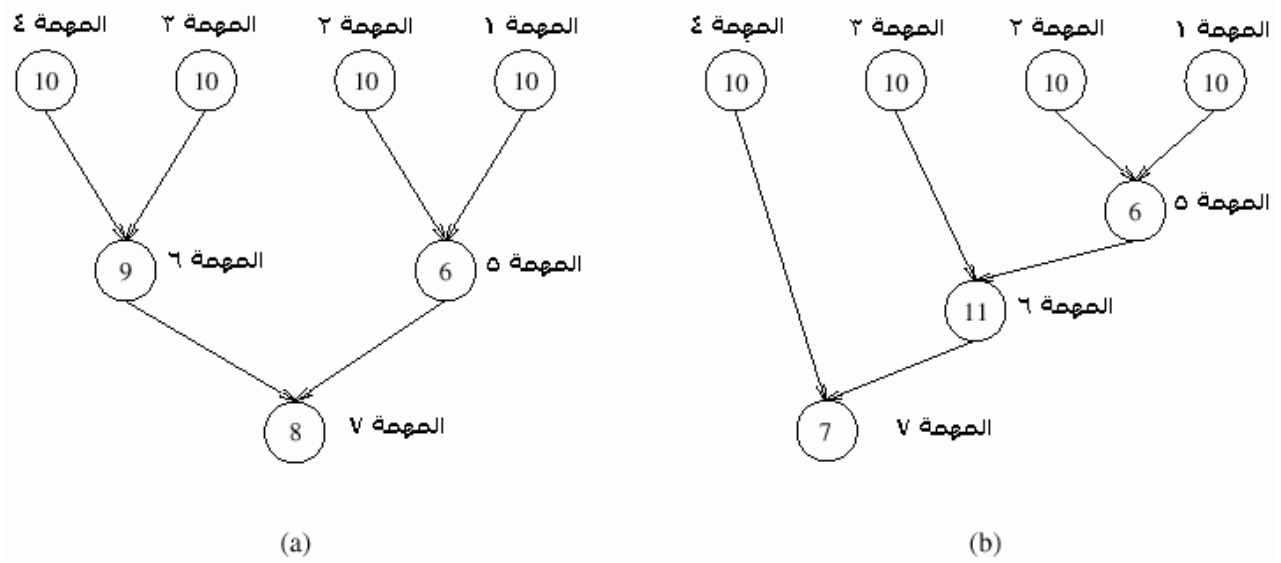
 ((3-2) (3-3))

 .

 2.33 (3-5.a)

 .

 1.88 (3-5.b)



الشكل (3-5): تجريد لمخططي التبعية للشكلين (3-2) و (3-3)

(3-5.b)

27

(3-5.a)

34

1.88 2.33

64 63

N^2

(3.1)

$O(N^2)$

• تفاعل المهمة (Task-Interaction):

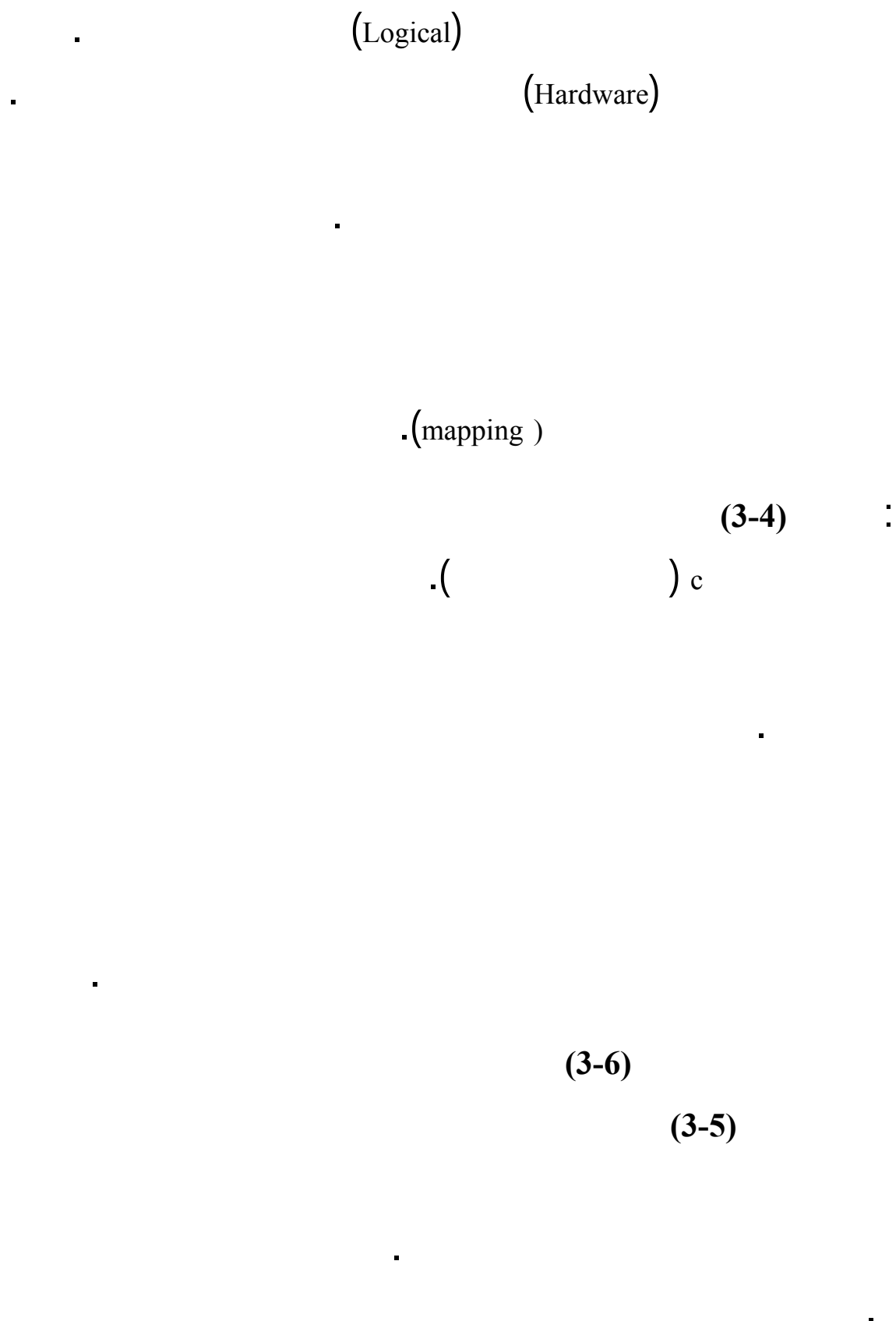
()

.b

b

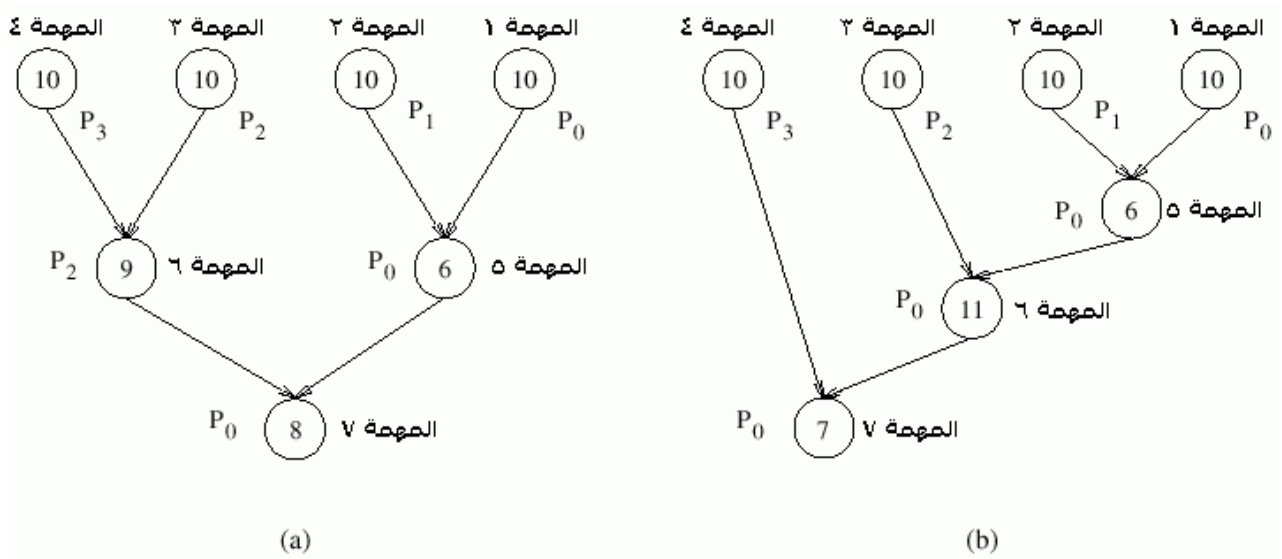
b

3.2 الإجراءات والمقابلة



(3-6.b) :

$P_1 \quad P_0$ $P_2 \quad 5$
 $.P_1 \quad P_0$ $.P_2$



الشكل (3-6) المقابلة لمخطط المهام في شكل (3-5) إلى أربعة إجراءات P .

3.3 تقنيات التقسيم

(Decomposition)

▪

▪

▪

▪

:

• (Recursive Decomposition)

• (Data Decomposition)

• (Exploratory Decomposition)

• (Speculative Decomposition)

▪

▪

3.3.1 التقسيم العَوْدِي (Recursive Decomposition)

(divide-and-conquer)" - "

مثال (3-3): الفرز السريع (Quicksort)

n A ()

.()

A X

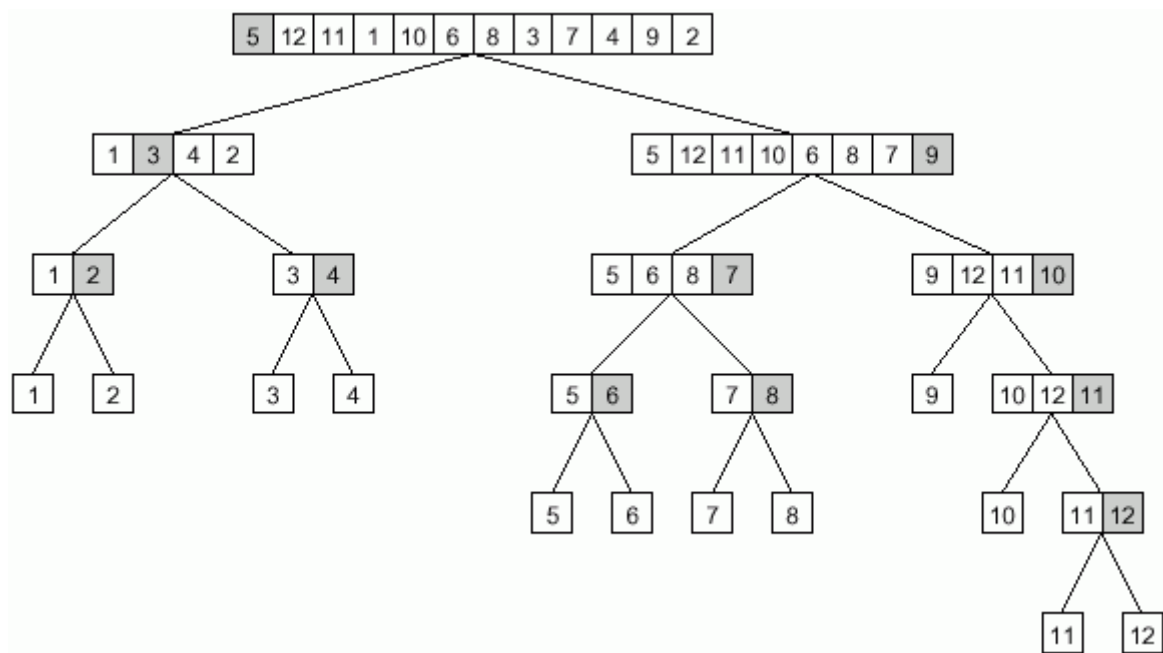
A1 X A0 A1 A0

.X

A1 A0

.Quicksort

. 12 (3-7)



الشكل (3-7): مخطط التبعية للفرز السريع والقائم على التقسيم العودي لتقسيم متسلسلة من ١٢ رقم.

(3-7)

(3-7)

.()

A1 A0)

(

.(-)

n

A

A

3-1.

الخوارزمية 3-1: برنامج تسلسلي لإيجاد العدد الأصغر في مصفوفة أعداد A بطول n .

```
1.  procedure SERIAL_MIN (A, n)
2.  begin
3.    min = A[0];
4.    for i := 1 to n - 1 do
5.      if (A[i] < min) min := A[i];
6.    endfor;
7.    return min;
8.  end SERIAL_MIN
```

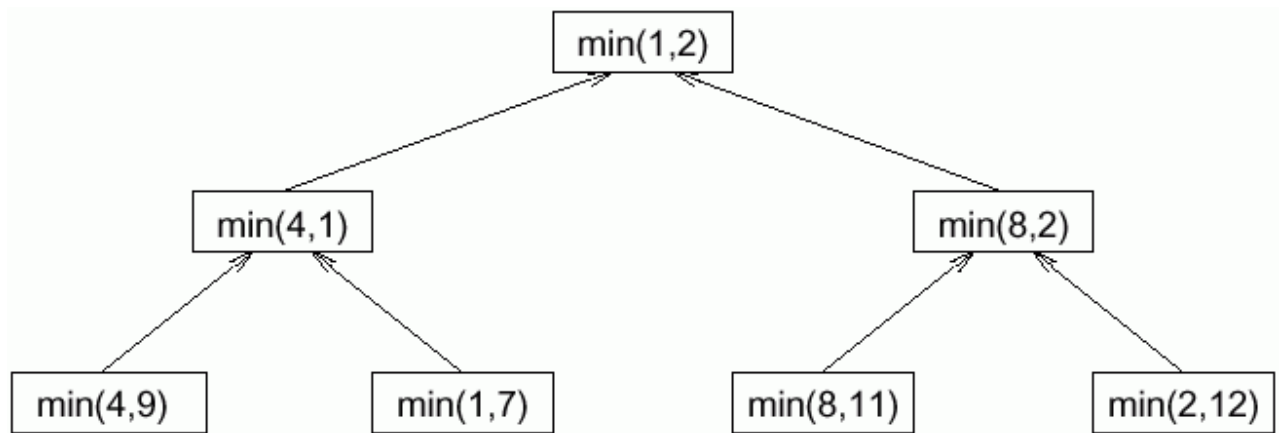
" - "

" - "

3-2

A

$(n/2)$



الشكل (3-8): مخطط التبعية لإيجاد العدد الأصغر للسلسلة [4,9,1,7,8,11,2,12]. كل عقدة في الشجرة تمثل مهمة لإيجاد العدد الأصغر من عددين.

الخوارزمية 2-3: برنامج عودي لإيجاد العدد الأصغر من بين عناصر A المكونة من n عدداً .

```

1.  procedure RECURSIVE_MIN (A, n)
2.  begin
3.    if (n = 1) then
4.      min := A[0];
5.    else
6.      lmin := RECURSIVE_MIN (A, n/2);
7.      rmin := RECURSIVE_MIN (&(A[n/2]), n - n/2);
8.      if (lmin < rmin) then
9.        min := lmin;
10.     else
11.       min := rmin;
12.     endelse;
13.  endelse;
14.  return min;
15.  end RECURSIVE_MIN
  
```

3.3.2 تقسيم البيانات (Data Decomposition)

.

:

.

.

.

.

•

.

(3-4)

.

.

المثال (3-4) ضرب المصفوفات المربعة

(B A)

(3-9)

.C

 $n \times n$

()

.

)

 $n/2 \times$

) C

.(

 $(n/2$

.B A

$$\begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \cdot \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix} \rightarrow \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix}$$

(a)

$$\text{Task 1: } C_{1,1} = A_{1,1}B_{1,1} + A_{1,2}B_{2,1}$$

$$\text{Task 2: } C_{1,2} = A_{1,1}B_{1,2} + A_{1,2}B_{2,2}$$

$$\text{Task 3: } C_{2,1} = A_{2,1}B_{1,1} + A_{2,2}B_{2,1}$$

$$\text{Task 4: } C_{2,2} = A_{2,1}B_{1,2} + A_{2,2}B_{2,2}$$

(b)

الشكل (3-9): (a) التجزيء لمصفوفات المدخلات والمخرجات إلى مصفوفات جزئية بحجم 2×2 .
(b) التقسيم لمسألة ضرب المصفوفات إلى أربع مهام اعتماداً على تجزيء المصفوفات الوارد في (a).

C

(3-9)

(3-10)

(3-9.a).

التقسيم I	التقسيم II
Task 1: $C_{1,1} = A_{1,1} B_{1,1}$	Task 1: $C_{1,1} = A_{1,1} B_{1,1}$
Task 2: $C_{1,1} = C_{1,1} + A_{1,2} B_{2,1}$	Task 2: $C_{1,1} = C_{1,1} + A_{1,2} B_{2,1}$
Task 3: $C_{1,2} = A_{1,1} B_{1,2}$	Task 3: $C_{1,2} = A_{1,2} B_{2,2}$
Task 4: $C_{1,2} = C_{1,2} + A_{1,2} B_{2,2}$	Task 4: $C_{1,2} = C_{1,2} + A_{1,1} B_{1,2}$
Task 5: $C_{2,1} = A_{2,1} B_{1,1}$	Task 5: $C_{2,1} = A_{2,2} B_{2,1}$
Task 6: $C_{2,1} = C_{2,1} + A_{2,2} B_{2,1}$	Task 6: $C_{2,1} = C_{2,1} + A_{2,1} B_{1,1}$
Task 7: $C_{2,2} = A_{2,1} B_{1,2}$	Task 7: $C_{2,2} = A_{2,1} B_{1,2}$
Task 8: $C_{2,2} = C_{2,2} + A_{2,2} B_{2,2}$	Task 8: $C_{2,2} = C_{2,2} + A_{2,2} B_{2,2}$

الشكل (3-10): مثالان لتقسيم عملية ضرب المصفوفة إلى ثمانية مهام.

(itemset)

.

المثال (3-5): حساب تكرارات المكونات في التعامل مع قاعدة البيانات

(itemsets)

.Transaction Database

I (Transaction) n T I T

(itemset) .itemset m

T .

.

I

.

(3-11.a)

(3-11)

Itemset

{D,E}

:

(a) Transactions (input), itemsets (input), and frequencies (output)

Database Transactions	A, B, C, E, G, H	Itemsets	A, B, C	Itemset Frequency	1
	B, D, E, F, K, L		D, E		3
	A, B, F, H, L		C, F, G		0
	D, E, F, H		A, E		2
	F, G, H, K,		C, D		1
	A, E, F, K, L		D, K		2
	B, C, D, G, H, L		B, C, F		0
	G, H, L		C, D, K		0
	D, E, F, K, L				
	F, G, H, L				

(b) Partitioning the frequencies (and itemsets) among the tasks

Database Transactions	A, B, C, E, G, H	Itemsets	A, B, C	Itemset Frequency	1
	B, D, E, F, K, L		D, E		3
	A, B, F, H, L		C, F, G		0
	D, E, F, H		A, E		2
	F, G, H, K,				
	A, E, F, K, L				
	B, C, D, G, H, L				
	G, H, L				
	D, E, F, K, L				
	F, G, H, L				

task 1

Database Transactions	A, B, C, E, G, H	Itemsets	C, D	Itemset Frequency	1
	B, D, E, F, K, L		D, K		2
	A, B, F, H, L		B, C, F		0
	D, E, F, H		C, D, K		0
	F, G, H, K,				
	A, E, F, K, L				
	B, C, D, G, H, L				
	G, H, L				
	D, E, F, K, L				
	F, G, H, L				

task 2

الشكل (3-11): حساب تكرار المكونات في تعاملات قاعدة البيانات

(3-11.b)

(3-11.b)

•

P

N

P

(N>P)

P

(3-5)

(3-12.a)

(a) Partitioning the transactions among the tasks

task 1			task 2		
Database Transactions	Itemsets	Itemset Frequency	Database Transactions	Itemsets	Itemset Frequency
A, B, C, E, G, H	A, B, C	1		A, B, C	0
B, D, E, F, K, L	D, E	2		D, E	1
A, B, F, H, L	C, F, G	0		C, F, G	0
D, E, F, H	A, E	1	A, E, F, K, L	A, E	1
F, G, H, K,	C, D	0	B, C, D, G, H, L	C, D	1
	D, K	1	G, H, L	D, K	1
	B, C, F	0	D, E, F, K, L	B, C, F	0
	C, D, K	0	F, G, H, L	C, D, K	0

(b) Partitioning both transactions and frequencies among the tasks

task 1			task 2		
Database Transactions	Itemsets	Itemset Frequency	Database Transactions	Itemsets	Itemset Frequency
A, B, C, E, G, H	A, B, C	1	A, B, C, E, G, H		
B, D, E, F, K, L	D, E	2	B, D, E, F, K, L		
A, B, F, H, L	C, F, G	0	A, B, F, H, L		
D, E, F, H	A, E	1	D, E, F, H		
F, G, H, K,			F, G, H, K,	C, D	0
				D, K	1
				B, C, F	0
				C, D, K	0

task 3			task 4		
Database Transactions	Itemsets	Itemset Frequency	Database Transactions	Itemsets	Itemset Frequency
	A, B, C	0	A, E, F, K, L		
	D, E	1	B, C, D, G, H, L	C, D	1
	C, F, G	0	G, H, L	D, K	1
A, E, F, K, L	A, E	1	D, E, F, K, L	B, C, F	0
			F, G, H, L	C, D, K	0

الشكل (3-12): بعض التقسيمات لحساب تكرار المكونات في تعاملات قاعدة البيانات

(3-12.b)

Transaction

frequencies

.4

2

3

1

3.3.3 التقسيم الاستكشافي (Exploratory Decomposition)

" (15-Puzzle) "

المثال (3-6): مسألة أحجية المربع

15 1

15

.4X4

(3-13)

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

(a)

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

(b)

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

(c)

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

(d)

الشكل (3-13): مسألة أحجية المربع توضيح الترتيب الأول (a) والترتيب النهائي (d) وتتابع الحركات من الترتيب الأول وحتى الترتيب النهائي.

3 2

4

()

:

.

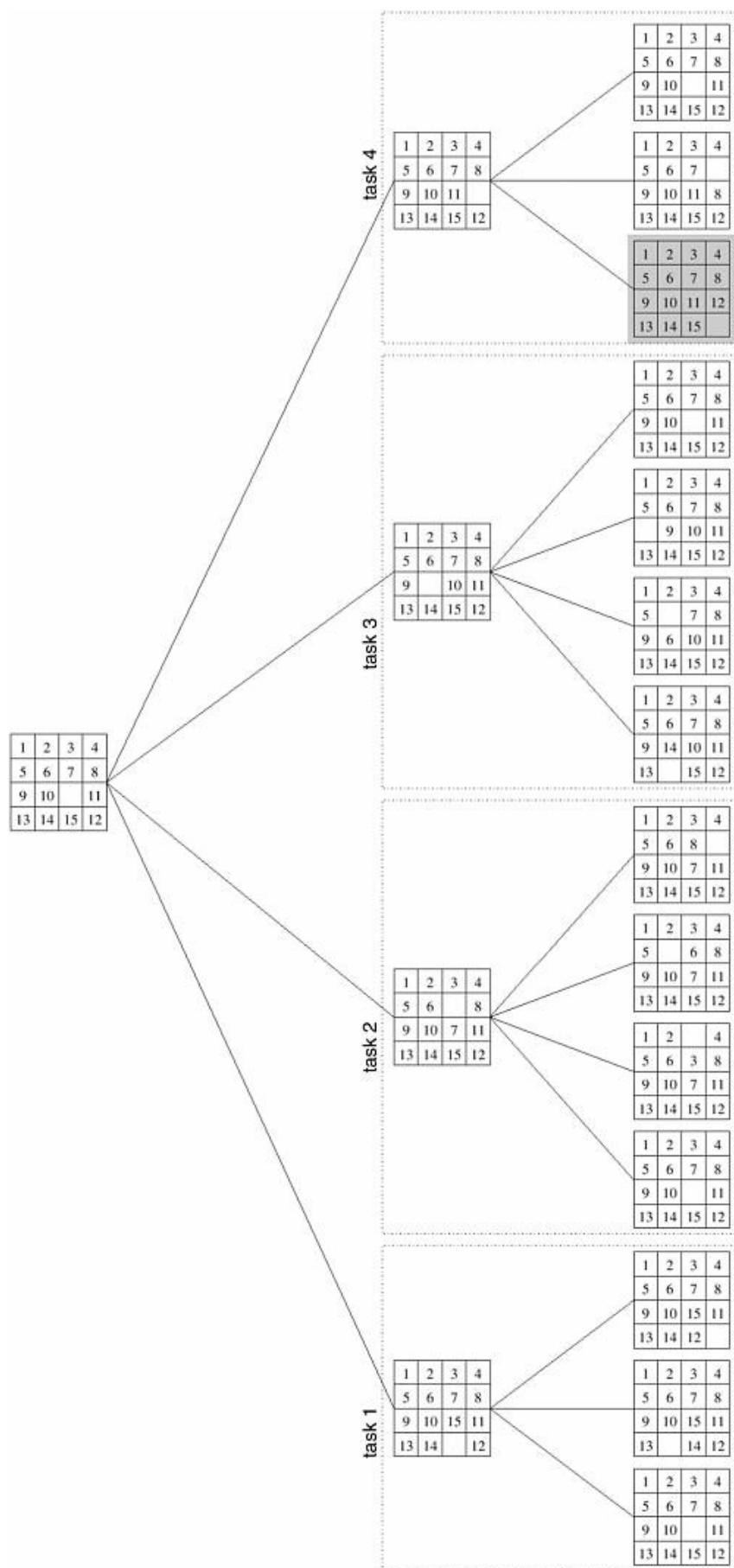
:

.

.

.

(3-14)



الشكل (3-14): الأوضاع التي تنتج عن مثال لمسألة أحجية المربع

)

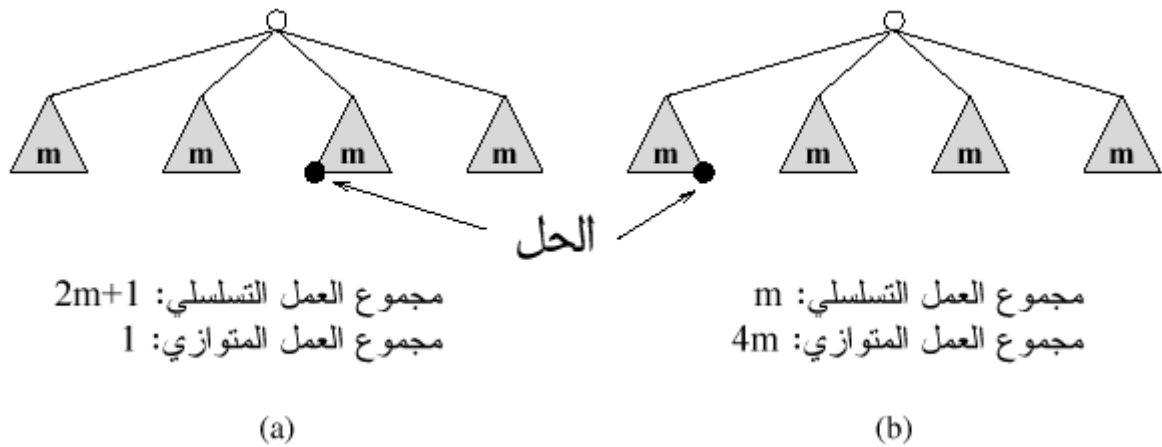
(

. (3-15)

((3-15.a)) 3

. 2 1

((3-15.b)) 1

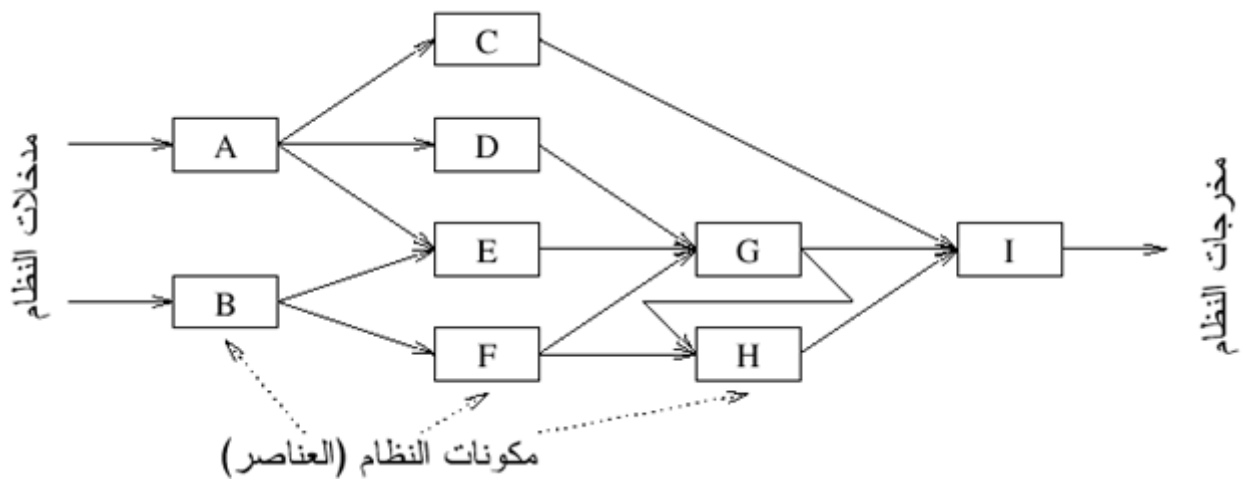


الشكل (3-15): توضيح للسرعة الغير منتظمة الناتجة عن التقسيم الاستكشافي.

3.3.4 التقسيم التخميني (Speculative Decomposition)

المثال (3-7): توازي محاكاة الحدث المتقطع

(3-16)



الشكل (3-16): شبكة مبسطة لمحاكاة الحدث المتقطع.

(3-7)

▪

)

▪

(

▪

,

▪

▪

-

▪

-

▪

▪

-

▪

-

▪

▪

3.3.5 التقسيم المختلط (Hybrid Decompositions)

—

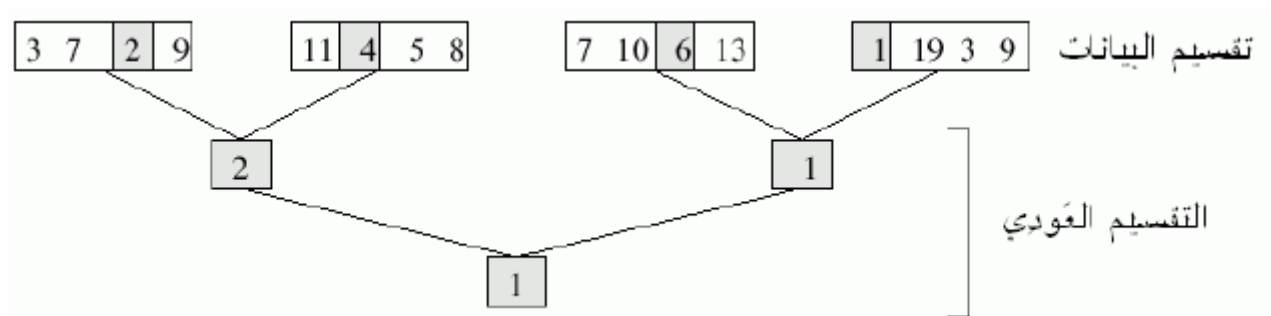
—

N
) P
P (

.

P

(3-17).



الشكل (3-17) التقسيم المختلط لإيجاد العدد الأصغر لمصفوفة من الحجم ١٦ باستخدام أربعة مهام.

.

(3-3)

.

$O(n)$

▪

 n

▪

 $O(n)$

▪

▪

3.4 أمثلة للخوارزميات المتوازية

· :

3.4.1 خوارزمية الفرز الفقاعي وتوابعها (Bubble Sort).

bubble)

$a_1, >$

(sort

:

"

-

"

$n-1$

$< a_2, \dots, a_n$

·

$\cdot (a_1, a_2), (a_2, a_3), \dots, (a_{n-1}, a_n)$

-

·

$n-1$

·

(3-3)

·

compare-exchange

·

$\Theta(n)$

$$\left(\begin{array}{c} \Theta(n) \\ \Theta(n^2) \end{array} \right)$$

$$5 \quad 4 \quad) \quad - \quad (3-3)$$

•
•

خوارزمية (3-3): خوارزمية الفرز الفقاعي التسلسلي.

```

1.  procedure BUBBLE_SORT(n)
2.  begin
3.      for i := n - 1 downto 1 do
4.          for j := 1 to i do
5.              compare-exchange(aj, aj + 1);
6.  end BUBBLE_SORT

```

3.4.1.1 الإبدال الزوجي-الفردى (Odd-Even Transposition)

$$\left(\begin{array}{c} n \\ n \end{array} \right) \quad " \quad - \quad " \quad n/2$$

$a_1, a_2, >$

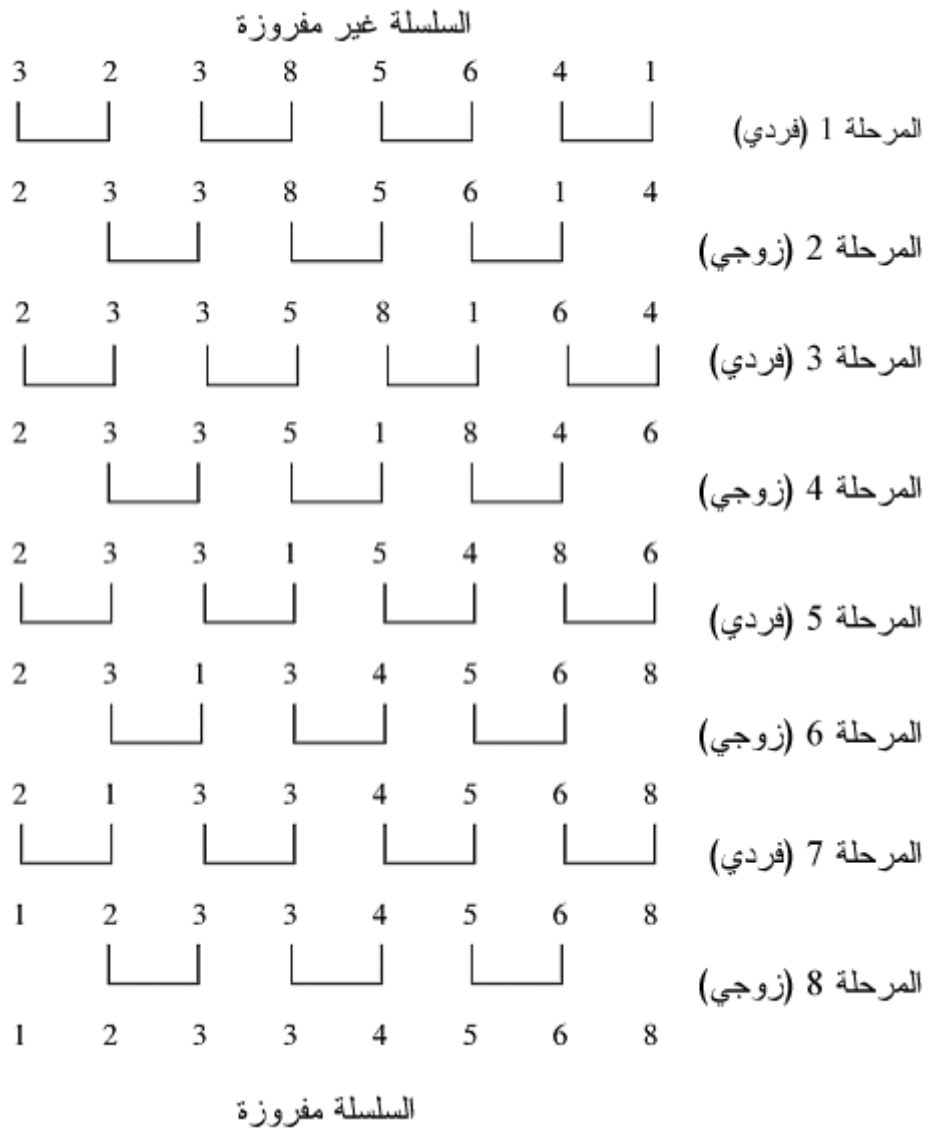
$< \dots, a_n$

$a_1, a_2,)$

$$\left(\begin{array}{c} n \end{array} \right) \quad - \quad ((a_3, a_4), \dots, (a_{n-1}, a_n)$$

$a_2, a_3), (a_4, a_5), \dots, (a_{n-2},$

$a_{n-1})$.
 n .
 $\Theta(n)$ ()
 $\Theta(n^2)$.
 (3-18)



الشكل (3-18): فرز 8 عناصر ($n=8$) باستخدام خوارزمية الإبدال الزوجى-الفردى، خلال كل مرحلة هناك 8 عناصر يتم مقارنتها ($n=8$)

الخوارزمية (3-4): الخوارزمية التسلسلية للإبدال الفردي-الزوجي.

```

1.  procedure ODD-EVEN(n)
2.  begin
3.    for i := 1 to n do
4.      begin
5.        if i is odd then
6.          for j := 0 to n/2 - 1 do
7.            compare-exchange(a2j + 1, a2j + 2);
8.          if i is even then
9.            for j := 1 to n/2 - 1 do
10.             compare-exchange(a2j, a2j + 1);
11.         end for
12.     end ODD-EVEN

```

الصيغة المتوازية لخوارزمية الإبدال الزوجي-الفردي:

-

-

n " " .

. (n)

. i= 1,2,3,...,n p_i a_i .

-

.

-

.(3-5) .

الخوارزمية (3-5): الصيغة المتوازية لخوارزمية الفرز بالإبدال الزوجي-الفردى، على عدد n -عملية بشكل حلقة.

```

1.  procedure ODD-EVEN_PAR (n)
2.  begin
3.      id := process's label
4.      for i := 1 to n do
5.          begin
6.              if i is odd then
7.                  if id is odd then
8.                      compare-exchange_min(id + 1);
9.                  else
10.                     compare-exchange_max(id - 1);
11.             if i is even then
12.                 if id is even then
13.                     compare-exchange_min(id + 1);
14.                 else
15.                     compare-exchange_max(id - 1);
16.             end for
17.  end ODD-EVEN_PAR

```

-

• n $\Theta(1)$ •

• $\Theta(n)$

$\Theta(n \log n)$ n

• $\Theta(n^2)$

-

p •

n/p

• $p < n$

• ()

3.4.2 خوارزمية بريم Prim لإيجاد أصغر شجرة هيكلية

(Graph Theory)

(Graph)

3.4.2.1 تعاريف ومفاهيم أساسية

V

G

$G=(V,E)$

(Graph)

E

(undirected graph)

(directed graph)

:

e

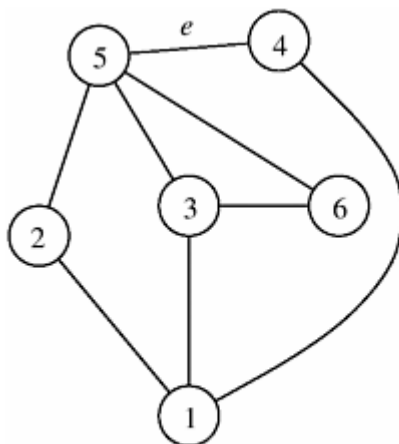
V

v u

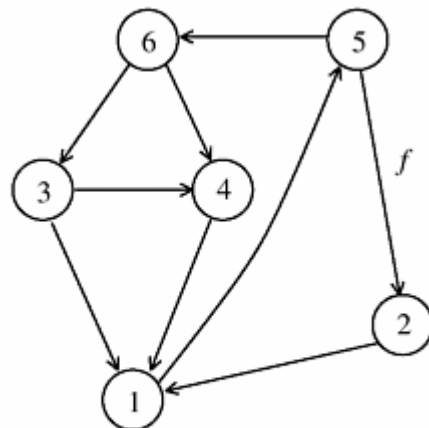
v u

(u,v)

.v u



(a)



(b)

الشكل (3-19) (a) بيان غير موجه، (b) بيان موجه.

$$a_{i,j} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases}$$

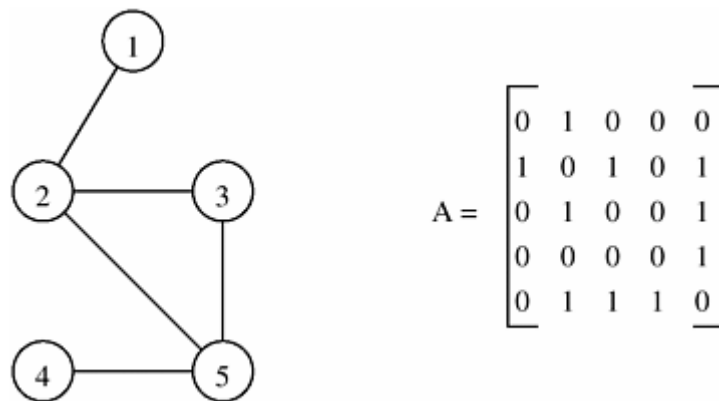
(3-20)

$$A = (a_{i,j})$$

(weighted graphs)

:

$$a_{i,j} = \begin{cases} w(v_i, v_j) & \text{if } (v_i, v_j) \in E \\ 0 & \text{if } i = j \\ \infty & \text{otherwise} \end{cases}$$



الشكل (3-20): بيان غير موجه وتمثيله بمصفوفة الجوار.

$$\Theta(n^2) \quad n$$

3.4.2.2 () :

Minimum Spanning Tree (MST): Prim's Algorithm

G

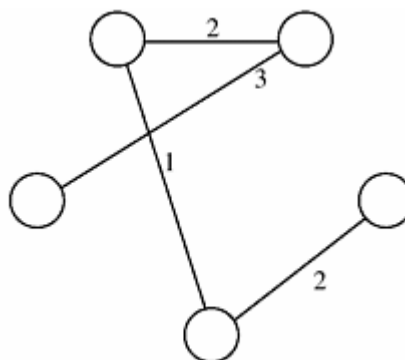
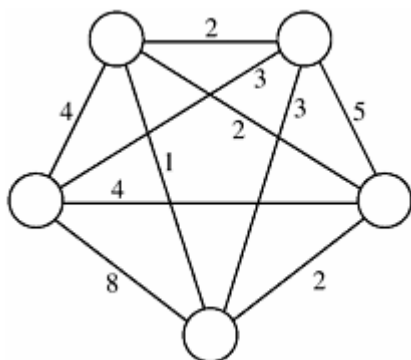
G

.G

(MST)

.

(3-21)



الشكل (3-21): بيان غير موجه، والشجرة الهيكلية الأصغر فيه.

G

G

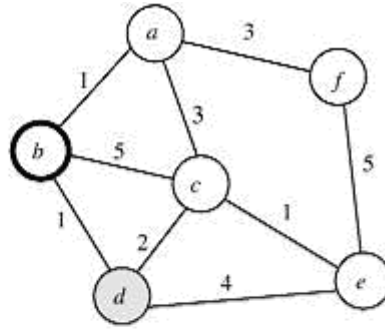
spanning forest

greedy)

.(algorithm

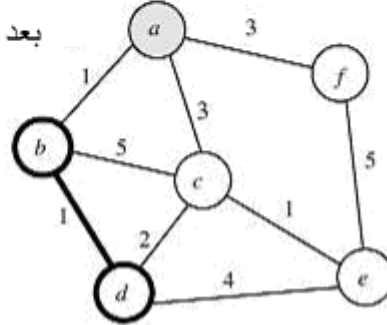
$$\begin{aligned}
 & \cdot \\
 & \cdot \\
 & G=(V,E,w) \\
 (3-6) \quad & \cdot G \quad A=(a_{i,j}) \\
 & V_T \quad \cdot \\
 & d[1..n] \quad \cdot \\
 & d[v] \quad (V - V_T) \quad v \\
 & \cdot v \quad V_T \\
 & \cdot \\
 & r \quad V_T \\
 & v \in (V - V_T), d[v] = w(r, v) \quad v \quad d[r] = 0 \\
 & \cdot d[v] = \infty \\
 & V_T \quad u \\
 & d[v] \quad \cdot d[u] = \min\{d[v] \mid v \in (V - V_T)\} \\
 & v \in (V - V_T) \\
 & \cdot V_T = V \quad \cdot u \quad v \\
 & (3-22) \\
 (3-6) \quad & \cdot \sum_{v \in V} d[v]
 \end{aligned}$$

(a) البيان الأصلي



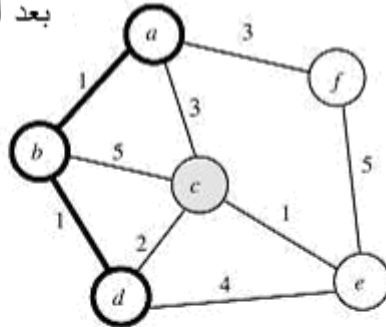
	a	b	c	d	e	f
d[]	1	0	5	1	∞	∞
a	0	1	3	∞	∞	3
b	1	0	5	1	∞	∞
c	3	5	0	2	1	∞
d	∞	1	2	0	4	∞
e	∞	∞	1	4	0	5
f	2	∞	∞	∞	5	0

(b) بعد اختيار الضلع الأول



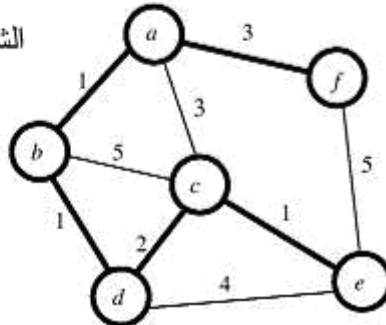
	a	b	c	d	e	f
d[]	1	0	2	1	4	∞
a	0	1	3	∞	∞	3
b	1	0	5	1	∞	∞
c	3	5	0	2	1	∞
d	∞	1	2	0	4	∞
e	∞	∞	1	4	0	5
f	2	∞	∞	∞	5	0

(c) بعد اختيار الضلع الثاني



	a	b	c	d	e	f
d[]	1	0	2	1	4	3
a	0	1	3	∞	∞	3
b	1	0	5	1	∞	∞
c	3	5	0	2	1	∞
d	∞	1	2	0	4	∞
e	∞	∞	1	4	0	5
f	2	∞	∞	∞	5	0

(d) الشجرة الهيكلية الأصغر



	a	b	c	d	e	f
d[]	1	0	2	1	1	3
a	0	1	3	∞	∞	3
b	1	0	5	1	∞	∞
c	3	5	0	2	1	∞
d	∞	1	2	0	4	∞
e	∞	∞	1	4	0	5
f	2	∞	∞	∞	5	0

الشكل (3-22): خوارزمية بريم لأصغر شجرة هيكلية. الجذر لأصغر شجرة هيكلية هو b. ومن أجل كل تكرار فيوضح بالخط الغامق الرؤوس التي في V_T وكذلك الأضلاع المختارة. المصفوفة $d[v]$ تعرض قيم الرؤوس في $V - V_T$ بعد أن يتم تحديثها.

$$n-1 \quad (10-13) \quad \text{while} \quad (3-6)$$

$$O(n) \quad (12-13) \quad \text{for} \quad (10) \quad \min\{d[v] | v \in (V - V_T)\}$$

$$\cdot \Theta(n^2)$$

الخوارزمية (3-6): خوارزمية بريم التسلسلية لأصغر شجرة هيكلية.

```

1. procedure PRIM_MST(V, E, w, r)
2. begin
3.    $V_T := \{r\};$ 
4.    $d[r] := 0;$ 
5.   for all  $v \in (V - V_T)$  do
6.     if edge  $(r, v)$  exists set  $d[v] := w(r, v);$ 
7.     else set  $d[v] := \infty;$ 
8.   while  $V_T \neq V$  do
9.     begin
10.      find a vertex  $u$  such that  $d[u] := \min\{d[v] | v \in (V - V_T)\};$ 
11.       $V_T := V_T \cup \{u\};$ 
12.      for all  $v \in (V - V_T)$  do
13.         $d[v] := \min\{d[v], w(u, v)\};$ 
14.      endwhile
15.    end PRIM_MST

```

• الصيغة المتوازية لخوارزمية بريم:

•

$$V_T \quad v \quad d[v]$$

$$c \quad d \quad b \quad (3-23)$$

$$d[c] \quad d$$

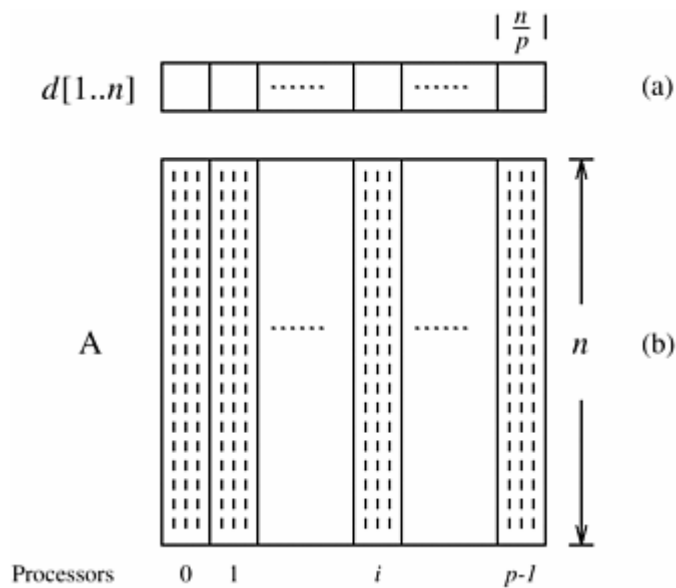
•

• While

•

n p
 n/p p V
 V_i
 d P_i $i=0,1,\dots,p-1$ P_i
(3-24.a) $\left(v \in V_i \quad d[v] \quad P_i \right) V_i$
while P_i
 $d_i[u] = \min \{ d_i[v] \mid v \in (V - V_T) \cap V_i \}$

P_0 P_0 $d_i[u]$
 u P_0 V_T u
 V_T u u P_i
 $\left(\quad \right) d[v]$



الشكل (3-24): تقسيم المصفوفة d ومصفوفة الجوار A إلى P إجرائية

$$\begin{aligned}
 & (V - V_T) \cdot v \cdot d[v] \cdot V_T \cdot u \\
 & \cdot (u, v) \cdot v \cdot \\
 & V_i \cdot P_i \\
 & \cdot \\
 & \cdot (3-24.b) \cdot \Theta(n^2/p)
 \end{aligned}$$

الفصل الرابع: البرمجة المتوازية

" :

"

:

"

"

.

.

البرمجة المتوازية (Parallel Programming): هي البرمجة بلغة تتضمن البنى أو الميزات المتوازية.

CSP(Communicating Sequential Process), OCCAM

Parallel-

.Parallel-C Parallel-Pascal FORTRAN

C/C++ FORTRAN

.Pthreads(POSIX Threads) PVM MPI

OCCAM

FORTRAN-90

.C++

MPI

4.1 لغة OCCAM

OCCAM

OCCAM

CSP(Communicating Sequential Process)

CSP

OCCAM

Process

OCCAM

(Concurrent Processes)

¹ الترانسبيوتر Transputer: اختصار للجملة **transistor computer**. وهو حاسب آلي متكامل على شريحة واحدة، يتضمن ذاكرة وصول عشوائي (RAM) و وحدة حسابية للنقطة-العائمة (FPU)، ويصمم هذا الجهاز أساساً كوحدة بناء لأنظمة الحاسب المتوازي.

OCCAM

.

OCCAM

:

:

SYNTAX: <variable>:=<expression>
Example: x :=y +1

"?"

:

SYNTAX: <channel>?<variable>
Example: Ch ? x

"!"

:

SYNTAX: <channel>!<expression>
Example: Ch !y +1

:

SYNTAX: SKIP
Example: SKIP

:

SYNTAX: STOP
Example: STOP

(Hierarchical)

.

.

OCCAM

PAR

.()

: S1,S2,S3

PAR /*Ececute the following in parallel */

S1

S2

S3

: PAR S1,S2

CHAN OF INT in,out,middle; /*Declaration of the channels */

PAR /*Execute in parallel*/

INT X: /*The first process*/

WHILE TRUE

SEQ /*Sequential execution of following*/

In ? X /*Read X from in channel*/

Middle ! X /*Write X on middle channel*/

INT X: /*The second process*/

WHILE TRUE

SEQ /*Sequential execution of following*/

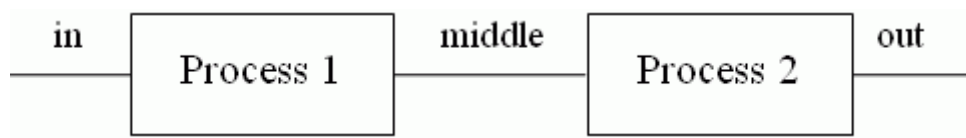
MIDDLE ? X /*Read X from in channel*/

OUT ! X /*Write X on out channel*/

buffer

(4-1)

Middle



الشكل (4-1)

PAR

:

PAR I=0 FOR N
S[I]

/*create n process and execute then in parallel*/

PAR

.

:

PAR

.

.

:

PLACED PAR

PLACED PAR

/*Place in parallel*/

PROCESSOR 1

/*Processor naming*/

P1

/*Task P1 on processor 1*/

PROCESSOR 2

/*Processor naming*/

P2

/*Task P2 on processor 2*/

PROCESSOR 3

/*Processor naming*/

P3

/*Task P3 on processor 3*/

PAR

OCCAM

.

SEQ

:OCCAM

.

:

.PAR

:

.

OCCAM

.

Type

.

:

CHAN OF <TYPE> <NAME1>,<NAME2>;

Example:

CHAN OF INT ch; /*ch: channel of type integer*/

OCCAM

Types

:

PROTOCOL int.real IS

INT;REAL32;

CHAN OF int.real ch:

INT inint;outint:

REAL32 inreal;outreal:

PAR

Ch ! outint;outreal

Ch ? inint;inreal

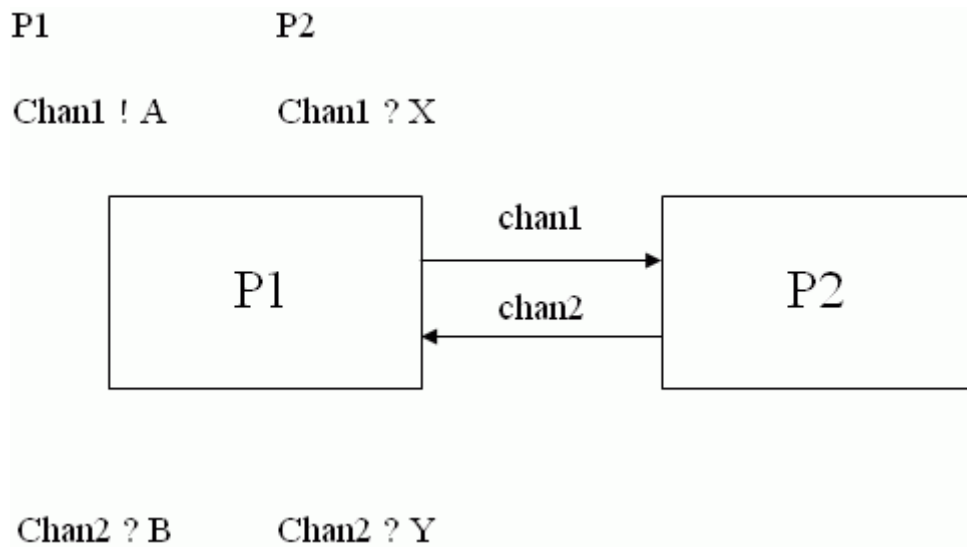
(!)

:

.()

.

4-) Chan1,Chan2 P1,P2 :
(2).



الشكل (4-2)

(Priority)

PRI

OCCAM

PRI

:PRI

PRI PAR

P1 /*highest priority*/

PAR

P2 /*three process with middle priority*/

P3

P4

P5 /*lowest priority*/

OCCAM

FOR, WHILE,...

▪

OCCAM

▪

.Processor1

▪

PLACED PAR	<i>/*Plase in parallel*/</i>
PROCESSOR 0	<i>/*Processor naming*/</i>
PLACE in AT link 0 in	<i>/*Place channel in on link 0 in(physical channel)*/</i>
PLACE out AT link 1 out	<i>/*Place channel out on link 1 out(physical channel)*/</i>
keyboard(in,out)	<i>/*Place process keybord on Processor 0*/</i>
PROCESSOR 1	<i>/*Processor naming Processor*/</i>
PLACE out 1 AT link 0 out	<i>/*Place channel out 1 on link 0 out*/</i>
PLACE in 1 AT link 0 in	<i>/*Place channel in 1 on link 1 in*/</i>
Screen(in 1, out 1)	<i>/*Place process screen on Processor 1*/</i>

4.2 لغة FORTRAN- 90 :

SIMD

FORTRAN

▪

:

N

for I=1,N

A[I]=B[I] + C[I]

:

A(1:N)=B(1:N) + C(1:N)

T(1:N)=A(2:N+1)

B(1:N)=2*T(1:N)

▪

▪

▪

HPF:Hight Performance FORTRAN

▪

FORTRAN-90

▪

▪

4.3 واجهة تمرير الرسائل MPI Message Passing Interface

()1994 MPI

.

.FORTRAN C/C++

MPI

.

125

MPI

(4-1)

6

.

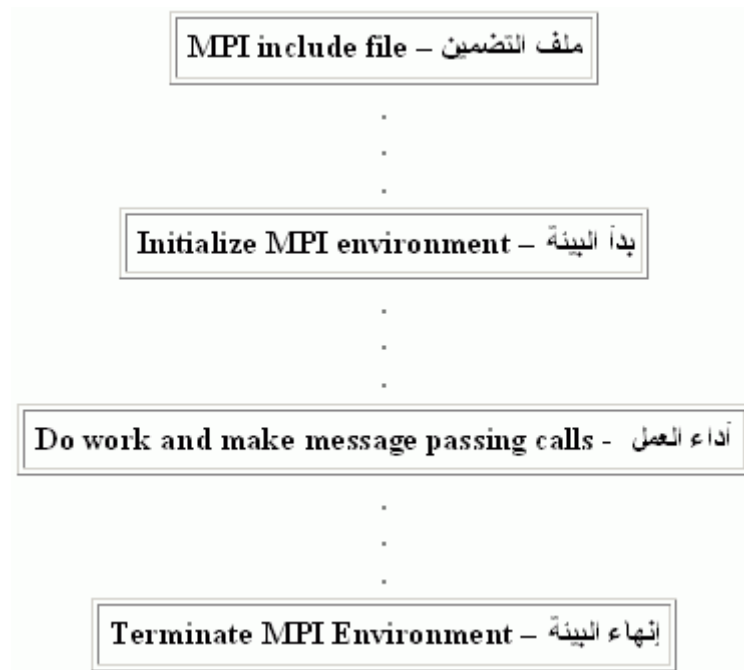
.MPI

الجدول (4-1): مجموعة من الروتينات الأساسية الخاصة بالمكتبة MPI

MPI_Init	بداية MPI
MPI_Finalize	إنهاء MPI
MPI_Comm_size	تحديد عدد المعالجات
MPI_Comm_rank	تحديد عنوان أو رتبة المعالج المستدعي
MPI_Send	إرسال رسالة
MPI_Recv	استقبال رسالة

4.3.1 الهيكل العام لبرامج MPI

:MPI



MPI_Init

MPI

.MPI

MPI_Finalize

.

MPI

MPI_Finalize MPI_init

.

:C++

```
int MPI_Init(int *argc, char ***argv)
int MPI_Finalize()
```


MPI_Finalize()

.MPI

"Welcome!"

4.3.2 المراسلات (Communicators)

MPI

.(communication domain)

MPI_Comm

MPI

MPI

MPI_COMM_WORLD

4.3.3 الحصول على معلومات عن بيئة التشغيل

MPI_Comm_rank MPI_Comm_size

:

```
int MPI_Comm_size(MPI_Comm comm, int *size)
```

```
int MPI_Comm_rank(MPI_Comm comm, int *rank)
```

Size

MPI_Comm_size

.comm

.rank

.

:

MPI_Comm_rank

.

rank

.rank

.

```
#include <iostream.h>
#include <mpi.h>

main(int argc, char *argv[])
{
    int npes, myrank;

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &npes);
    MPI_Comm_rank(MPI_COMM_WORLD, &myrank);

    cout <<"Welcome!  from process " <<myrank;
    cout <<"of " <<npes <<endl;
    MPI_Finalize();
}
```

```

Welcome!  from process 0 of 4
Welcome!  from process 2 of 4
Welcome!  from process 3 of 4
Welcome!  from process 1 of 4
```

4.3.4 تراسل البيانات في MPI

:

MPI

.

MPI_Recv

MPI_Send

.

:

```

int MPI_Send(
    void*          message /*in*/,
    int            count  /*in*/,
    MPI_Datatype    datatype /*in*/,
    int            dest    /*in*/,
    int            tag     /*in*/,
    MPI_Comm        comm   /*in*/
)

int MPI_Recv(
    void*          message /*out*/,
    int            count  /*in*/,
    MPI_Datatype    datatype /*in*/,
    int            source  /*in*/,
    int            tag     /*in*/,
    MPI_Comm        comm   /*in*/,
    MPI_Status*    status  /*out*/
)

```

:

.

(buffer)

• message -

.

• count -

- - datatype
- - source (rank)
- - dest
- - tag
- - comm
- - status

MPI_Send (buffer) .buf
 .datatype
 MPI_Send .count
 (rank) dest .comm dest
 .comm
 tag
 .MPI_TAG_UP MPI tag

أنواع البيانات في MPI:

C++

MPI

MPI

(4-2).

.MPI_PACKED MPI_BYTE

الجدول(4-2): يوضح أنواع البيانات في C++، ويوضح ما يقابلها في MPI.

أنواع البيانات في MPI	أنواع البيانات في C++
MPI_CHAR	signed char
MPI_SHORT	signed short int
MPI_INT	signed int
MPI_LONG	signed long int
MPI_UNSIGNED_CHAR	unsigned char
MPI_UNSIGNED_SHORT	unsigned short int
MPI_UNSIGNED	unsigned int
MPI_UNSIGNED_LONG	unsigned long int
MPI_FLOAT	Float
MPI_DOUBLE	Double
MPI_LONG_DOUBLE	long double
MPI_BYTE	-----
MPI_PACKED	-----

MPI_Recv

.comm

source

tag

.

tag

source

MPI

MPI_ANY_SOURCE

source

tag

.

MPI_ANY_TAG

.

datatype

count

.buf

.

MPI_Recv

.

```

        .
        .MPI_ERR_TRUNCATE

        status

MPI_Status.        status        C++        .

        :

typedef struct MPI_Status {
    int MPI_SOURCE;
    int MPI_TAG;
    int MPI_ERROR;
};

        MPI_TAG    MPI_SOURCE

MPI_ANY_SOURCE        .

        MPI_ERROR        .MPI_ANY_TAG

        .

        status

        status

        :        MPI_Get_count

int MPI_Get_count(MPI_Status *status, MPI_Datatype datatype,
                  int *count)

        status

        MPI_Recv        datatype

        .        count        MPI_Get_count
```

:

```

int  mynode,  totalnodes;
int  datasize;      // عدد وحدات المعطيات التي ستُرسل أو تستقبل
int  sender;        // رقم الإجراءية المرسل
int  receiver       // رقم الإجراءية المستقبلة
int  tag            // عدد صحيح يستخدم كقلب أو علامة للرسالة

MPI_Status status;  // متغير يحتوي معلومات عن الحالة

MPI_Init(&argc, &argv);
MPI_Comm_size(MPI_COMM_WORLD, &totalnodes);
MPI_Comm_rank(MPI_COMM_WORLD, &mynode);

// datasize تحديد عدد وحدات المعطيات
double * databuffer = new double[datasize];

// Fill in sender, receiver, tag on sender/receiver processes,
// and fill in databuffer on the sender process.

if (mynode==sender)
    MPI_Send(databuffer,datasize,MPI_DOUBLE,receiver,
             tag,MPI_COMM_WORLD);

if (mynode==receiver)
    MPI_Recv(databuffer,datasize,MPI_DOUBLE,sender,tag,
             MPI_COMM_WORLD,&status);

// انتهت عملية الإرسال والاستقبال

```

4.3.5 برامج تطبيقية باستخدام MPI

MPI

.

برنامج لإرسال واستقبال المعطيات

P0

.(P0)

P0

```

01 #include<iostream.h>
02 #include<mpi.h>
03
04 int main(int argc, char *argv[]){
05     int i;
06     int nitems =10;
07     int mynode, totalnodes;
08     MPI_Status status;
09
10     double *array;
11
12     MPI_Init(&argc, &argv);
13     MPI_Comm_size(MPI_COMM_WORLD, &totalnodes);
14     MPI_Comm_rank(MPI_COMM_WORLD, &mynode);
15
16     array =new double[nitems];
17
18     if(mynode ==0){
19         for(i=0; i<nitems; i++)
20             array[i]=(double) i;
21     }
22
23     if(mynode==0)
24         for(i=1; i<totalnodes; i++)
25             MPI_Send(array, nitems, MPI_DOUBLE, i,
26                        1, MPI_COMM_WORLD);
27     else
28         MPI_Recv(array, nitems, MPI_DOUBLE,
29                  0, 1, MPI_COMM_WORLD, &status);
30
31     for(i=0; i<nitems; i++){
32         cout <<"Processor " <<mynode;
33         cout <<":array[" <<i <<"]=" <<array[i]<<endl;
34     }
35
36     delete[] array;
37

```

```

36     MPI_Finalize();
37 }

```

تحليل البرنامج:

- MPI (12 14).
- (16)
- P0 (mynode=0)
- (18 21).
- P0 MPI_Send (totalnodes-1).
- (23 25)
- P0 MPI_Recv
- (26 27).
- (29)
- (32).
- (34).

برنامج إرسال المعطيات ضمن حلقة (Ring)

```

)
. + + + (
.

```

```

#include "mpi.h"
#include <iostream.h>

/*Set up communication tags (these can be anything)*/
const int to_right=201;

```

```
const int to_left=102;

void main (int argc,char *argv[])
{
    int value,new_value,procnum,numprocs;
    int right,left;
    int sum,i;

    MPI_Status recv_status;

    /*Initialize MPI */
    MPI_Init(&argc,&argv);

    /*Find out this processor number */
    MPI_Comm_rank(MPI_COMM_WORLD,&procnum);
    /*Find out the number of processors */
    MPI_Comm_size(MPI_COMM_WORLD,&numprocs);

    /*Compute number of the processor to the right */
    right =procnum +1;
    if (right ==numprocs) right =0;

    /*Compute number of the processor to the left */
    left =procnum -1;
    if (left ==-1)left =numprocs-1;

    sum =0;
    value =procnum;

    for(i =0;i <numprocs;i++){

        /*Send to the right */
        MPI_Send(&value,1,MPI_INT,right,to_right,
                MPI_COMM_WORLD);

        /*Receive from the left */
        MPI_Recv(&new_value,1,MPI_INT,left,to_right,
                MPI_COMM_WORLD,&recv_status);

        /*Sum the new value */
        sum =sum +new_value;

        /*Update the value to be passed */
        value =new_value;

        /*Print out the partial sums at each step */
        cout<<"PE "<<procnum<<": Partial sum ="<<sum<<endl;
    }

    /*Print out the final result */
    if (procnum ==0){
        cout<< "Sum of all processor numbers =" << sum << endl;
    }
    /*Shut down MPI */
    MPI_Finalize();
}
```

```
    return;  
}
```

إذا شغل البرنامج السابق على جهاز بأربع معالجات، فسيكون له الخرج التالي:

```
PE 1: Partial sum =0  
PE 2: Partial sum =1  
PE 3: Partial sum =2  
PE 0: Partial sum =3  
PE 1: Partial sum =3  
PE 2: Partial sum =1  
PE 3: Partial sum =3  
PE 0: Partial sum =5  
PE 1: Partial sum =5  
PE 2: Partial sum =4  
PE 3: Partial sum =3  
PE 0: Partial sum =6  
PE 1: Partial sum =6  
PE 2: Partial sum =6  
PE 3: Partial sum =6  
PE 0: Partial sum =6  
Sum of all processor numbers =6
```

برنامج جمع سلسلة أعداد

```
      :  
      .  
  
#include<iostream.h>  
  
int main(int argc,char **argv)  
{  
    int sum;  
  
    sum =0;  
  
    for(int i=1;i<=1000;i=i+1)  
        sum  = sum +  i;
```

```

    cout << "The sum from 1 to 1000 is:" << sum << endl;
}

```

)

.(

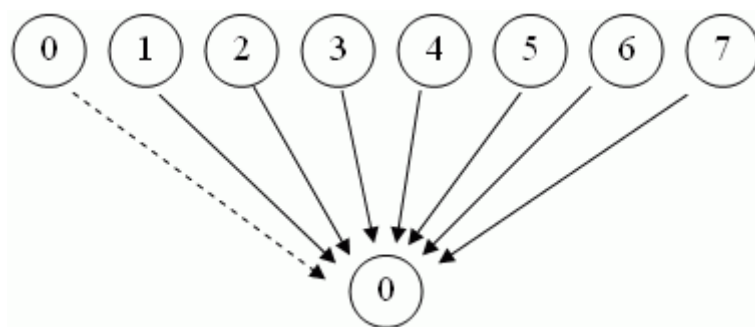
1

1000 501 500

.1000 1

P (P=8 : (4-3)) P (4-3)

P0



الشكل (4-3): تتجمع كل المعلومات إلى معالج واحد باستخدام الإرسال والاستقبال

:MPI

```

        (MPI_Comm_size
mynode      .(MPI_Comm_rank
totalnodes      MPI_Comm_rank
                .MPI_Comm_size
                :

startval = 1000*mynode/totalnodes+1;
endval = 1000*(mynode+1)/totalnodes;

mynode =0      totalnodes =1
                .endval =1000      startval=1
mynode =0      .1  0      mynode      totalnodes =2
endval      startval =501      mynode =1      endval =500      startval =1
                1000      .=1000
                )      (!      )
                .      (
                .endval      startval      for
                (      )
                .
                :      C++/MPI

#include<iostream.h>
#include<mpi.h>

```

```
int main(int argc, char **argv)
{
    int mynode, totalnodes;
    int sum, startval, endval, accum;
    MPI_Status status;

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &totalnodes); //get totalnodes
    MPI_Comm_rank(MPI_COMM_WORLD, &mynode); //get mynode

    sum = 0; //zero sum for accumulation
    startval = 1000 * mynode / totalnodes + 1;
    endval = 1000 * (mynode + 1) / totalnodes;

    for(int i = startval; i <= endval; i = i + 1)
        sum = sum + i;

    if(mynode != 0)
        MPI_Send(&sum, 1, MPI_INT, 0, 1, MPI_COMM_WORLD);

    else
    for(int j = 1; j < totalnodes; j = j + 1)
    {
        MPI_Recv(&accum, 1, MPI_INT, j, 1, MPI_COMM_WORLD,
                &status);
        sum = sum + accum;
    }
    if(mynode == 0)
        cout << "The sum from 1 to 1000 is:" << sum << endl;
    MPI_Finalize();
}
```

برنامج الفرز الزوجي-الفردى

-

"

.

"

```
void ODD_EVEN(int n){
    int i,j,temp;

    for(i=1;i<=n;i=i+1)
    {
        if(i%2==1){
            for(j=0;j<=n/2-1;j=j+1)
                if(a[2*j]>a[2*j+1])
                {
                    temp=a[2*j];
                    a[2*j]=a[2*j+1];
                    a[2*j+1]=temp;
                }
        }
        else{
            for(j=1;j<=n/2-1;j=j+1)
                if(a[2*j-1]>a[2*j])
                {
                    temp=a[2*j-1];
                    a[2*j-1]=a[2*j];
                    a[2*j]=temp;
                }
        }
    }
}
```

▪

⋮

```
#include <stdlib.h>
#include <mpi.h> /* Include MPI's header file */

main(int argc, char *argv[])
{
    int n;           /* The total number of elements to be sorted */
    int npes;        /* The total number of processes */
    int myrank;      /* The rank of the calling process */
    int nlocal;      /* The local number of elements, and the array that stores
them */
    int *elmnts;     /* The array that stores the local elements */
    int *relmnts;    /* The array that stores the received elements */
    int oddrank;     /* The rank of the process during odd-phase communication */
    int evenrank;    /* The rank of the process during even-phase communication */
    int *wspace;     /* Working space during the compare-split operation */
    int i;
    MPI_Status status;
```

```

/* Initialize MPI and get system information */
MPI_Init(&argc, &argv);
MPI_Comm_size(MPI_COMM_WORLD, &npes);
MPI_Comm_rank(MPI_COMM_WORLD, &myrank);

n = atoi(argv[1]);
nlocal = n/npes; /* Compute the number of elements to be stored locally. */

/* Allocate memory for the various arrays */
elmnts = (int *)malloc(nlocal*sizeof(int));
relmnts = (int *)malloc(nlocal*sizeof(int));
wspace = (int *)malloc(nlocal*sizeof(int));

/* Fill-in the elmnts array with random elements */
srandom(myrank);
for (i=0; i<nlocal; i++)
    elmnts[i] = random();

/* Sort the local elements using the built-in quicksort routine */
qsort(elmnts, nlocal, sizeof(int), IncOrder);

// Determine the rank of the processors that myrank needs to communicate
//during the odd and even phases of the algorithm */
if (myrank%2 == 0) {
    oddrank = myrank-1;
    evenrank = myrank+1;
}
else {
    oddrank = myrank+1;
    evenrank = myrank-1;
}

/* Set the ranks of the processors at the end of the linear */
if (oddrank == -1 || oddrank == npes)
    oddrank = MPI_PROC_NULL;
if (evenrank == -1 || evenrank == npes)
    evenrank = MPI_PROC_NULL;

/* Get into the main loop of the odd-even sorting algorithm */
for (i=0; i<npes-1; i++) {
    if (i%2 == 1) /* Odd phase */
        MPI_Sendrecv(elmnts, nlocal, MPI_INT, oddrank, 1, relmnts,
            nlocal, MPI_INT, oddrank, 1, MPI_COMM_WORLD, &status);
    else /* Even phase */
        MPI_Sendrecv(elmnts, nlocal, MPI_INT, evenrank, 1, relmnts,
            nlocal, MPI_INT, evenrank, 1, MPI_COMM_WORLD, &status);

    CompareSplit(nlocal, elmnts, relmnts, wspace,
        myrank < status.MPI_SOURCE);
}

free(elmnts); free(relmnts); free(wspace);
MPI_Finalize();
}

/* This is the CompareSplit function */
CompareSplit(int nlocal, int *elmnts, int *relmnts, int *wspace,
    int keepsmall)
{
    int i, j, k;

```

```
for (i=0; i<nlocal; i++)
    wspace[i] = elmnts[i]; /* Copy the elmnts array into the wspace array */

if (keepsmall) { /* Keep the nlocal smaller elements */
    for (i=j=k=0; k<nlocal; k++) {
        if (j == nlocal || (i < nlocal && wspace[i] < relmnts[j]))
            elmnts[k] = wspace[i++];
        else
            elmnts[k] = relmnts[j++];
    }
}
else { /* Keep the nlocal larger elements */
    for (i=k=nlocal-1, j=nlocal-1; k>=0; k--) {
        if (j == 0 || (i >= 0 && wspace[i] >= relmnts[j]))
            elmnts[k] = wspace[i--];
        else
            elmnts[k] = relmnts[j--];
    }
}
}

/* The IncOrder function that is called by qsort is defined as follows */
int IncOrder(const void *e1, const void *e2)
{
    return (*(int *)e1) - (*(int *)e2);
}
```



أهم المراجع

- [1] Daniel C.Hyde. Introduction to the Principles of Parallel Computation. Bucknell University, 1998
- [2] Daniel C.Hyde. Introduction to the Programming Language Occam. Bucknell University, 1995
- [3] Jim Buyens, Microsoft frontPage Version 2002 Inside Out, MSPress, 2001
- [4] Karniadakis, G., and Kirby II, R., Parallel Scientific Computing in C ++and MPI. Cambridge University Press, 2003.
- [5] Kumar, V , Gupta, A., Karypis, G., and Grama, A., Introduction to Parallel Computing, Second Edition. Addison Wesley Publishers, 2003.
- [6] Message Passing Interface (MPI),
<http://www.mhpcc.edu/training/workshop/mpi/MAIN.html>
- [7] Microsoft Computer Dictionary, Fifth Edition. MSPress, 2002.
- [8] Mikhail J. Atallah. Algorithms and Theory of Computation Handbook. CRC Press, 1998
- [9] Null, L., and Lobur, J., The Essentials of Computer Organization and Architecture. Jones and Bartlett Publishers, 2003.
- [10] Parhami, B., Introduction to Parallel Processing Algorithms and Architectures. KLUWER ACADEMIC PUBLISHERS, 2002
- [11] Richard Jenkins. Supercomputers of today and tomorrow. TAB BOOKS Inc., 1986.
- [12] TOP500 Supercomputer site, <http://www.top500.org/>.
- [13] Tutorial on Message Passing Interface Programming MPI(MPICH for Windows)
- [14] Message Passing Interface (MPI) Programming Lab
<http://www.dhpc.adelaide.edu.au/education/CS7933/lab/MPI/MPIlab.html>
- [15] معجم مصطلحات الكمبيوتر، الدار العربية للعلوم، 2001.
- [16] الربيعي وآخرون. المعجم الشامل لمصطلحات الحاسب الآلي والإنترنت. مكتبة العبيكان، 2001.