توازي خط الأنابيب **Pipelining**

ماهو توازي خط الأنابيب
What is Pipelining?

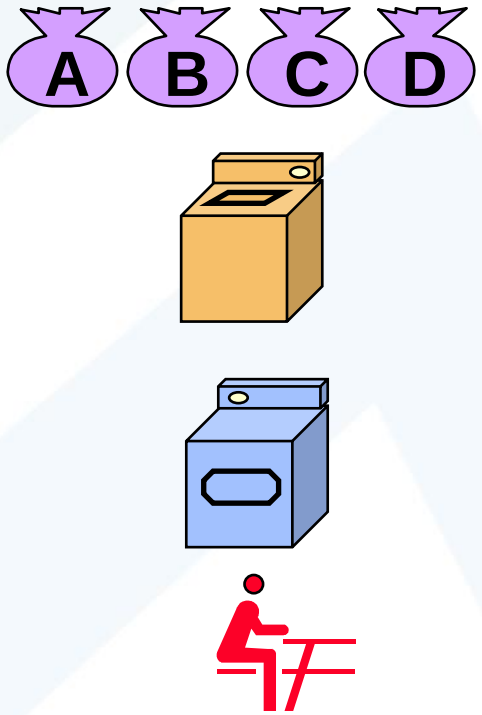
التوازي خط الأنابيب pipelining *هو تقنية تطبيق مفتاحي تستخدم من أجل بناء معالجات سريعة. تسمح هذه التقنية بالتراكب الزمني التنفيذي لعدة تعليمات.*

يشبه توازي السير المتحرك في المعالج خطوط تجميع السيارات ونطلق على كل محطة تجميع تسمية مرحلة أنبوبية *pipe stage* *أو قطاع أنبوبي pipe segment*.

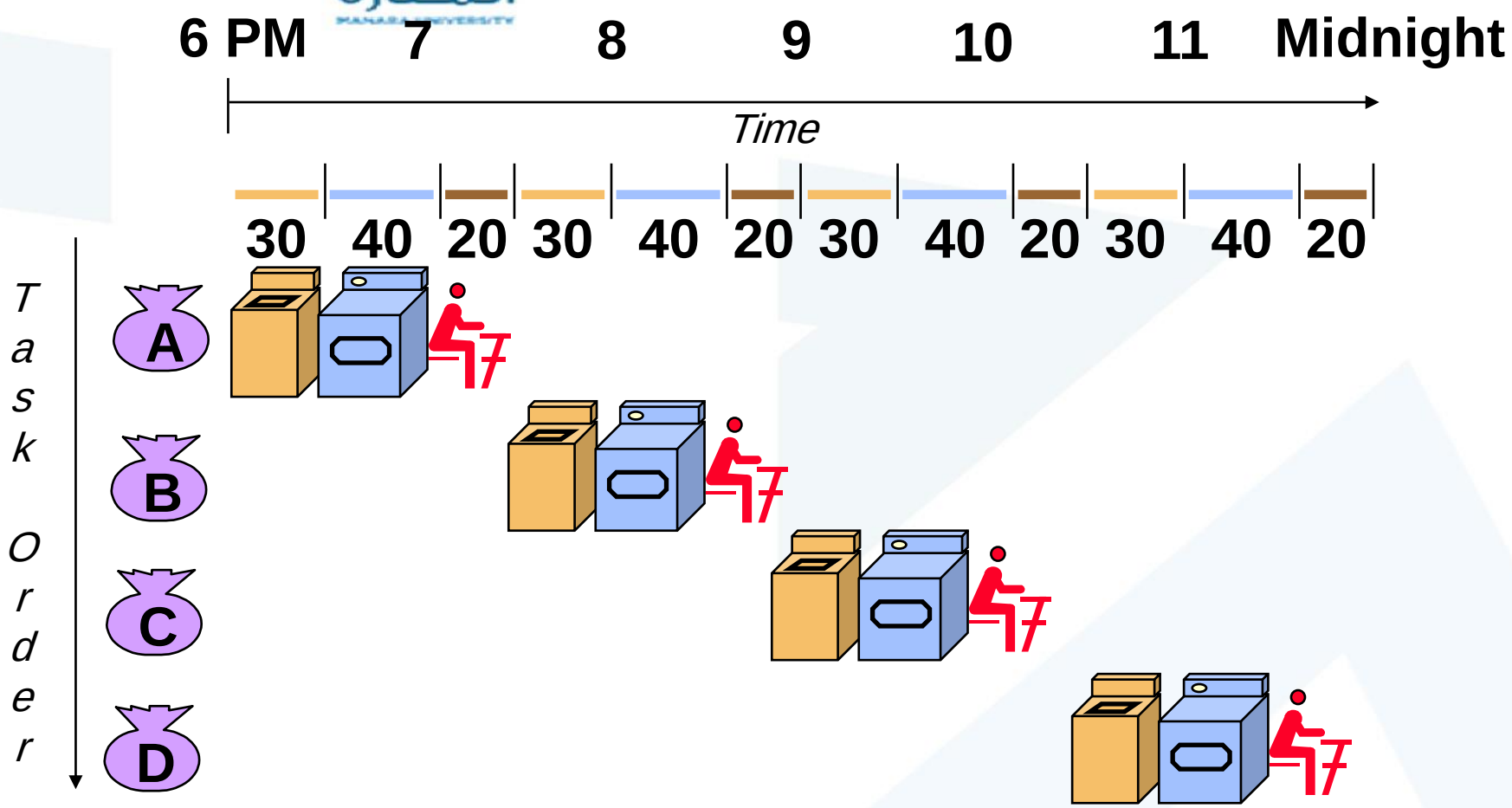
تقاس فعالية التمرير throughput لتعليمة أنبوبية بمقدار تكرار خروج التعليمة من خط الأنابيب.

طبيعة توازي خط الأنابيب
Pipelining: Its Natural!

- مثال حجرة غسل الملابس
- يحمل كل من Ann, Brian, Cathy, Dave مجموعة ملابس للغسيل wash والتجفيف dry والطوي fold
  - تستغرق مرحلة الغسيل 30 minutes
  - تستغرق مرحلة التجفيف 40 minutes
  - تستغرق مرحلة الطوي 20 minutes

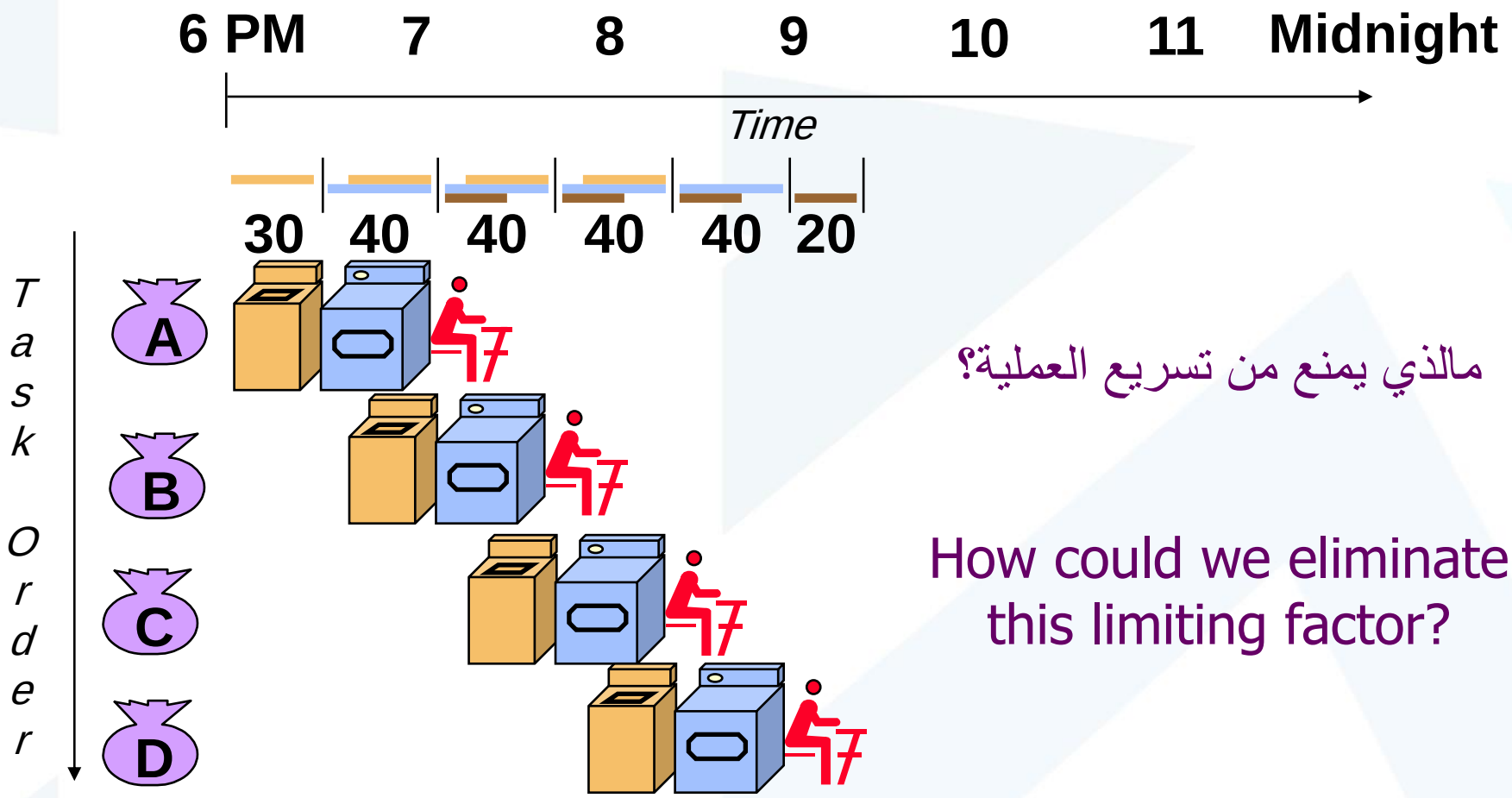


Sequential Laundry



- Sequential laundry takes 6 hours for 4 loads
- If they learned pipelining, how long would laundry take?

توازي أنبوبي

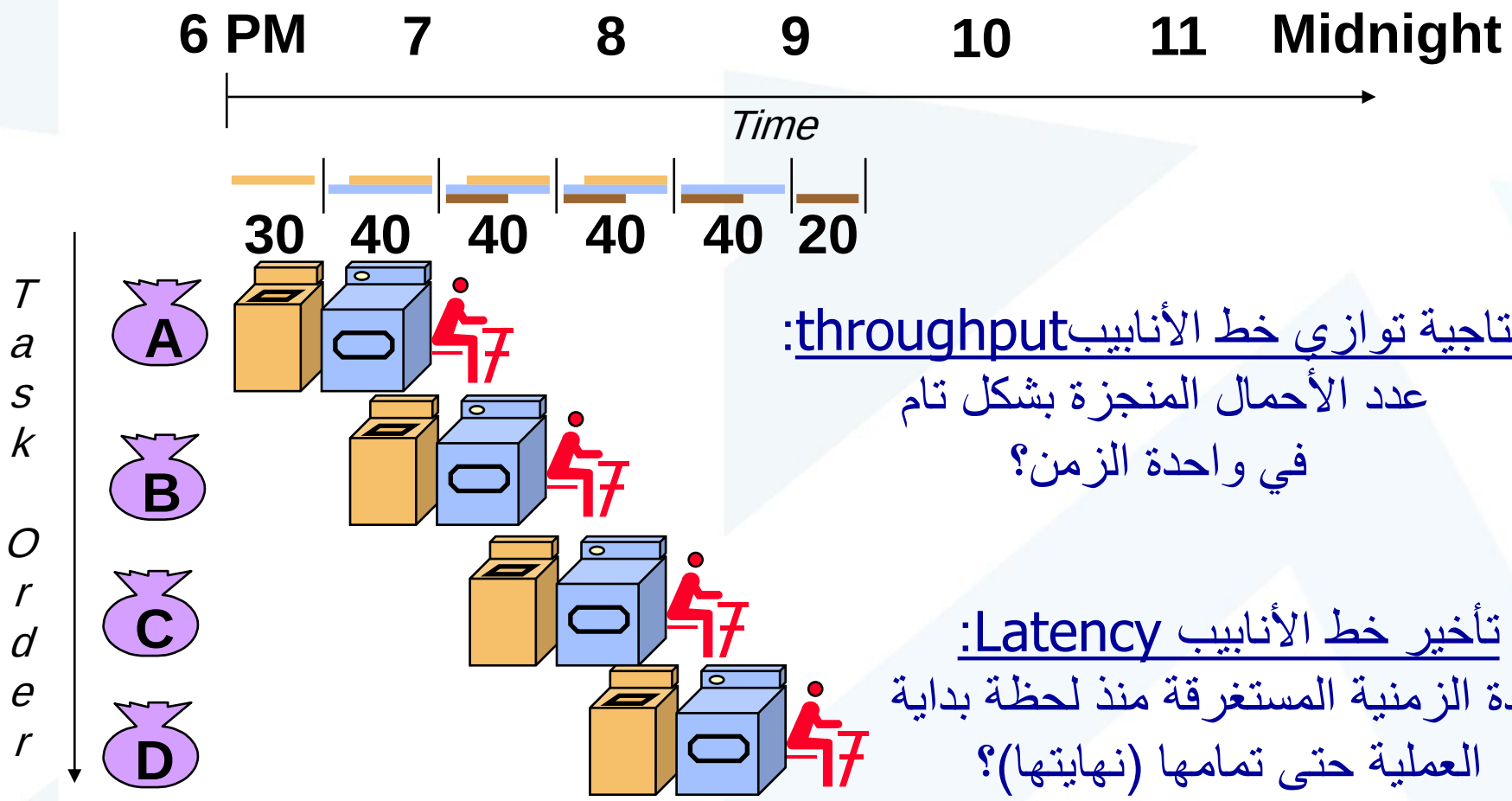


مالذي يمنع من تسريع العملية؟

How could we eliminate this limiting factor?

٣,٥ ساعة لأربعة أحمال

توازي أنبوبي

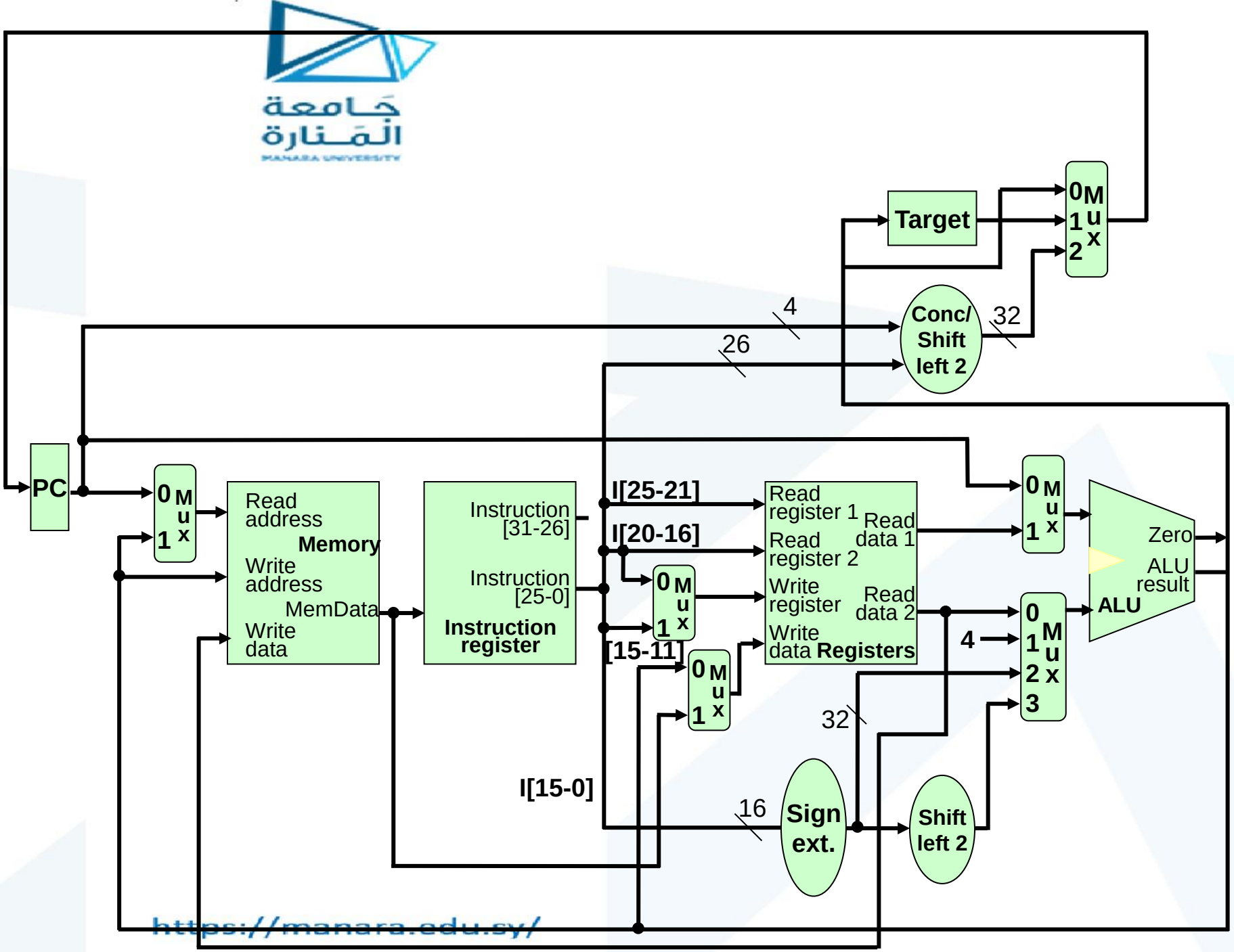


إنتاجية توازي خط الأنابيب throughput:

عدد الأحمال المنجزة بشكل تام في واحدة الزمن؟

تأخير خط الأنابيب Latency:

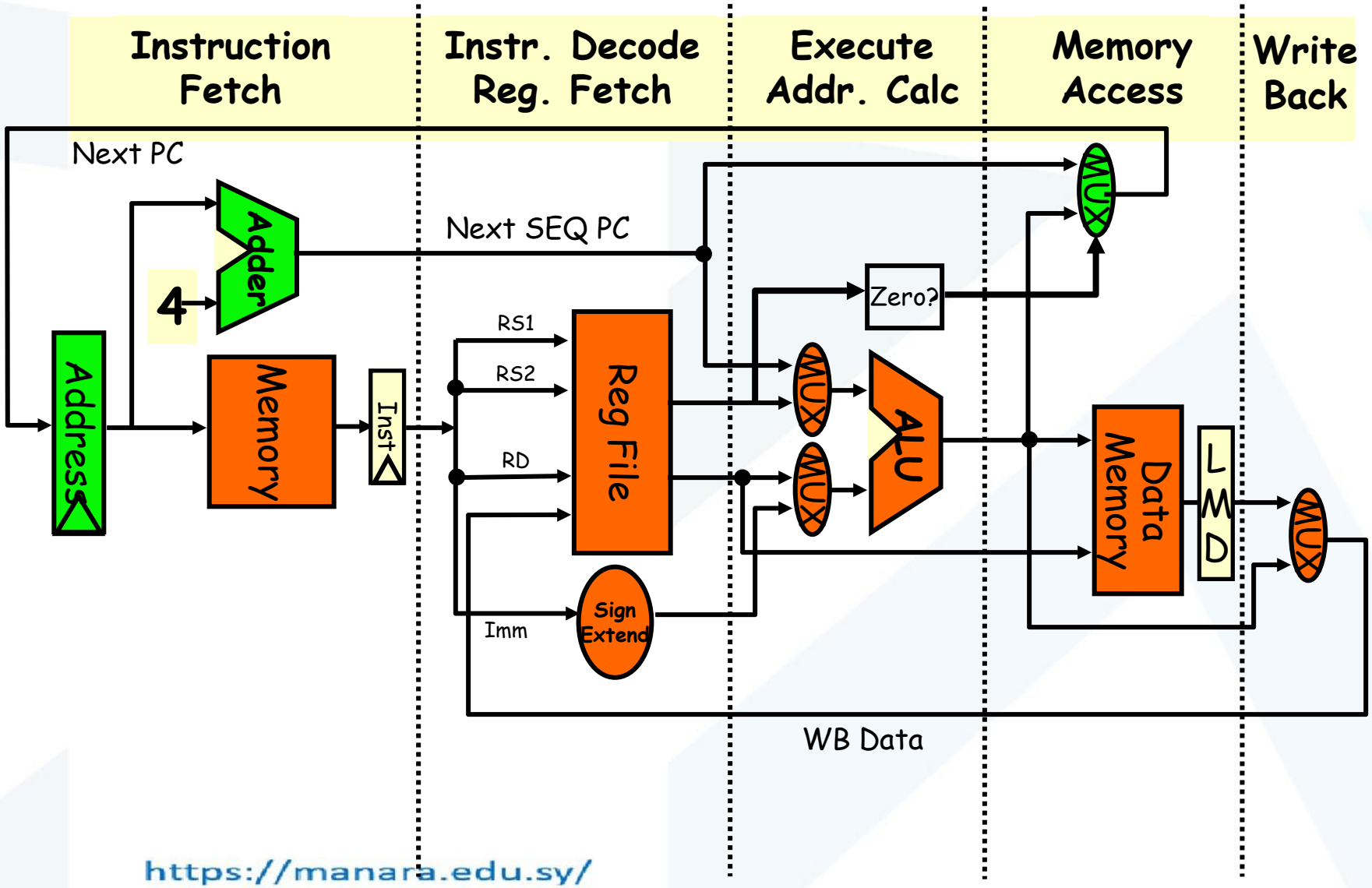
المدة الزمنية المستغرقة منذ لحظة بداية العملية حتى تمامها (نهايتها)؟



https://manara.edu.sy/

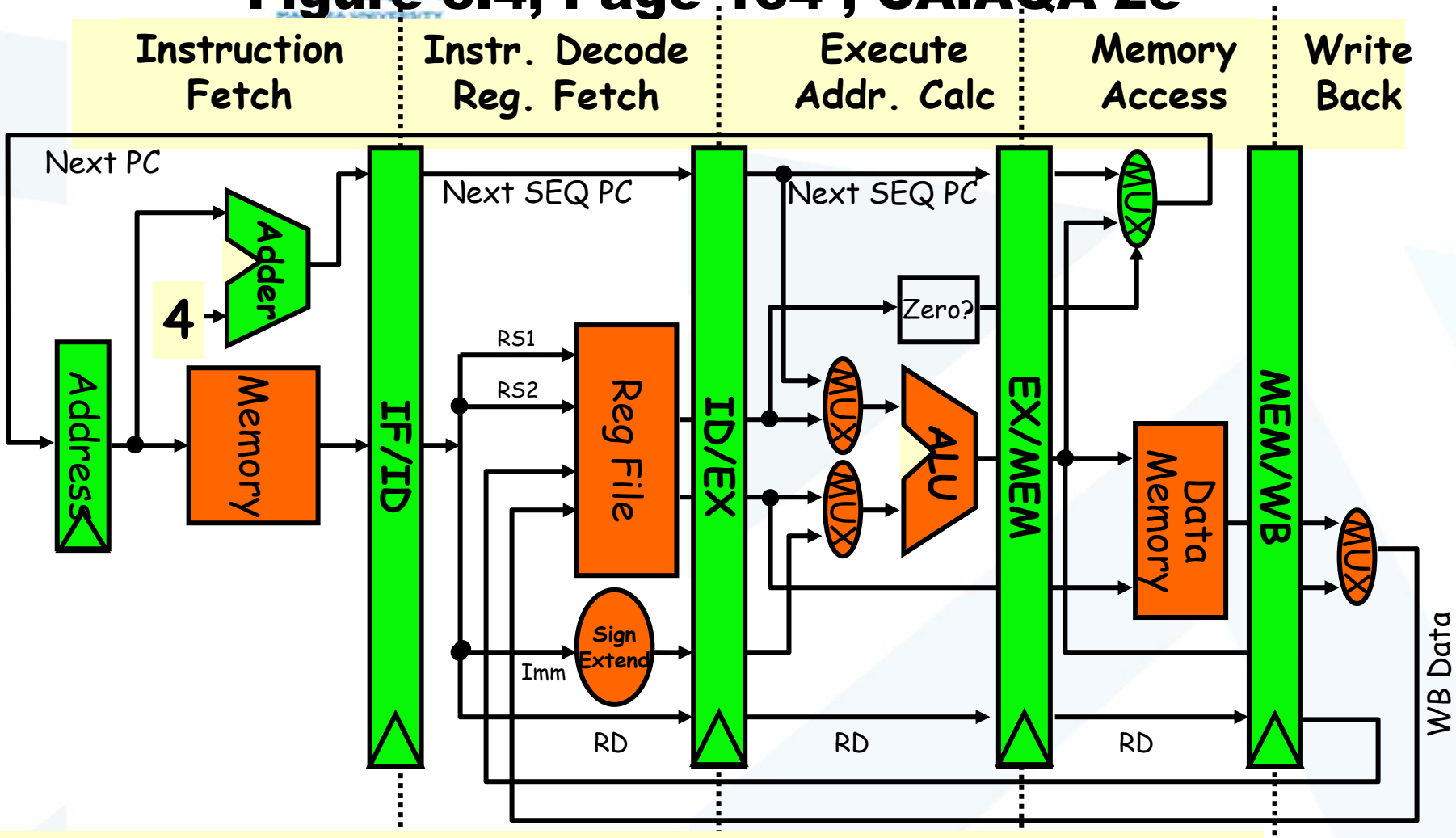
المراحل الخمس التي تمر بها البيانات في حواسب MIPS

5 Steps of MIPS Datapath



5 Steps of MIPS Datapath

Figure 3.4, Page 134 , CA:AQA 2e



- التحكم الساكن بالبيانات Data stationary control
  - فك ترميز محلي لكل مرحلة من مراحل التعليمة/ مرحلة التوازي الأنبوبي
  - local decode for each instruction phase / pipeline stage

Steps to Execute Each Instruction Type

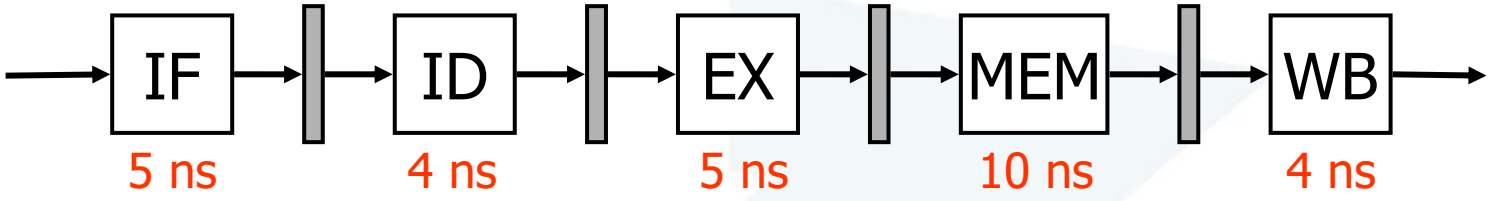
Step	Instruction Type				
	R-type	load	store	branch	jump
Fetch	IR ← Memory[PC] PC ← PC + 4				
Decode	A ← Registers[IR[25-21]] B ← Registers[IR[20-16]] Target ← PC + (sign-extend(IR[15-0]) << 2)				
Execute	ALUopt ← A op B	ALUout ← A + sign-extend(IR[15-0])		If(A==B) then PC←Target	PC ← concat(PC[31-28], IR[25-0]) << 2
Memory	Reg(R[15-11]) ← ALUout	Memdata ← Mem[ALUout]	Mem[ALUout] ← B		
Write-back		Reg(IR[20-16]) ← memdata			

مراحل توازي خط الأنابيب
Pipeline Stages

يمكننا تقسيم تنفيذ التعليمة إلى المراحل التالية:

IF: Instruction Fetch
ID: Instruction Decode
EX: Execution
MEM: Memory Access
WB: Write Back

تأخير وخرج توازي خط الأنابيب
Throughput and Latency

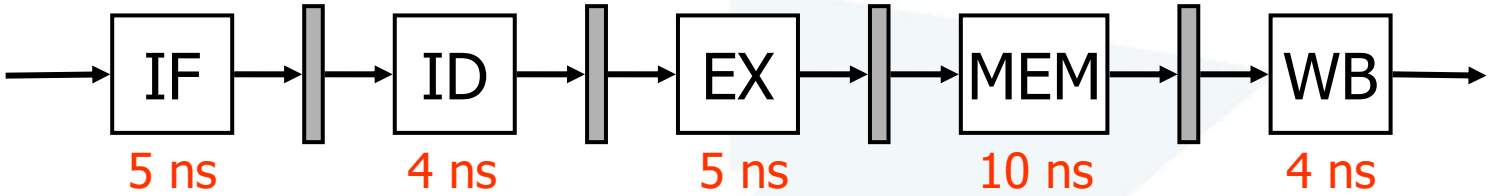


نريد أن نحدد من أجل توازي خط الأنابيب بأزمان التأخير المذكورة أعلاه ماهي إنتاجية التوازي الأنبوبي *pipeline throughput* وماهو التأخير الأنبوبي *pipeline latency*.

يقاس Pipeline throughput بعدد التعليمات المنتهية في الثانية

يقاس Pipeline latency بالمدة المستغرقة لتنفيذ تعليمة وحيدة.

تأخير وخرج توازي خط الأنابيب
Throughput and Latency



يقيس تأخير المرحلة الأنبوبية Pipeline throughput تواتر تنفيذ الأنبوب للتعليمات.

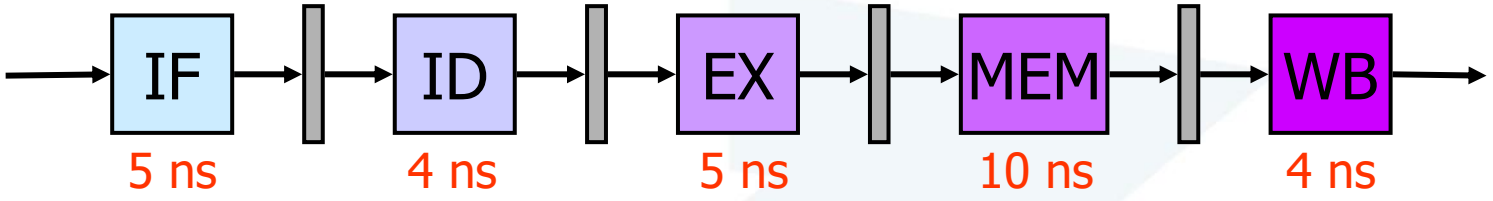
$$T = 1instr / \max[lat(IF), lat(ID), lat(EX), lat(MEM), lat(WB)]$$
$$= 1instr / \max[5ns, 4ns, 5ns, 10ns, 4ns]$$
$$= 1instr / 10ns$$

Pipeline latency تأخير التوازي الأنبوبي: يقيس الفترة الزمنية اللازمة لإتمام تنفيذ تعليمة بالتوازي الأنبوبي:

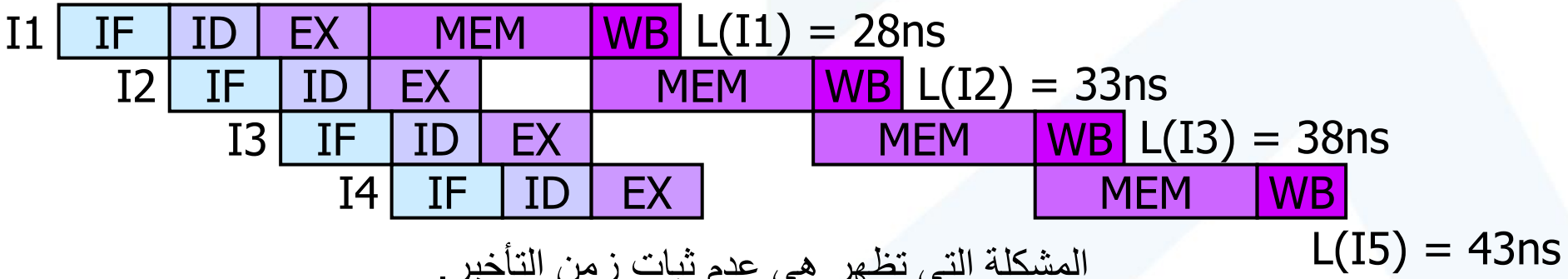
$$L = lat(IF) + lat(ID) + lat(EX) + lat(MEM) + lat(WB)$$
$$= 5ns + 4ns + 5ns + 10ns + 4ns = 28ns$$

Is this right?

تأخير وخرج توازي خط الأنابيب
Throughput and Latency



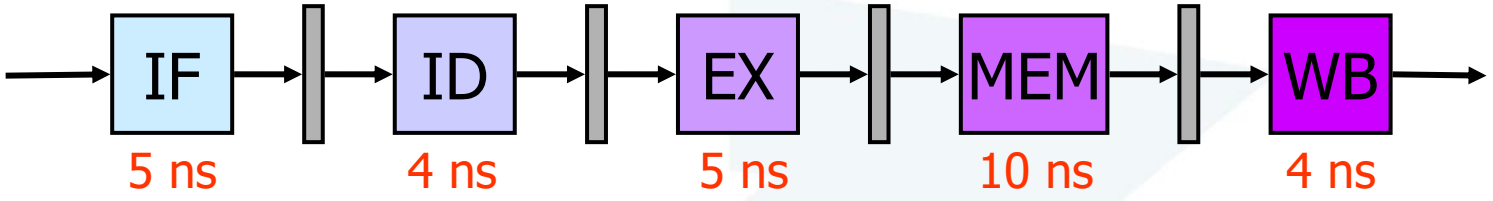
نضيف التأخيرات الزمنية لكل مرحلة فنحصل على التأخير الكلي للتعليمة.
يحسب التأخير لكل تعليمة على حدة



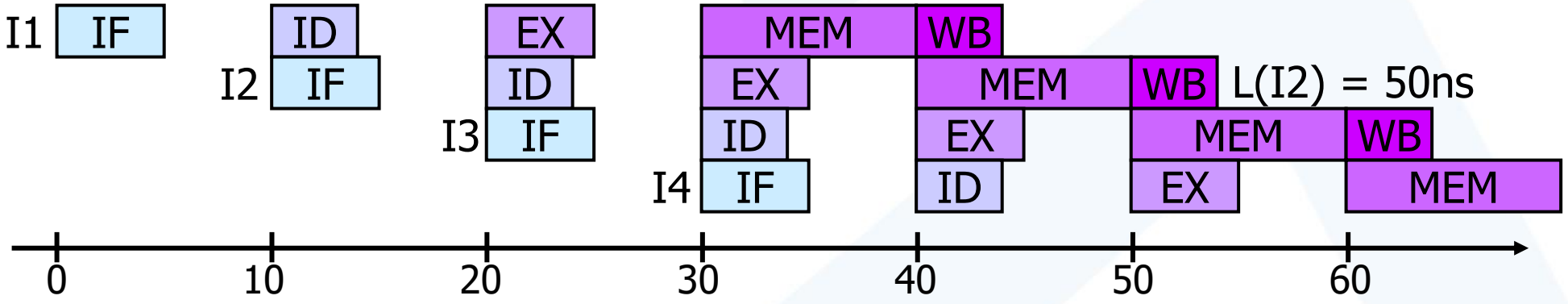
المشكلة التي تظهر هي عدم ثبات زمن التأخير.
ينجم ذلك عن عدم تساوي تأخير المراحل
والحل هو بجعل جميع المراحل تأخذ نفس الزمن وذلك
بجعل زمن كل مرحلة يساوي زمن المرحلة الأطول,

تأخير وخرج توازي خط الأنابيب

Throughput and Latency

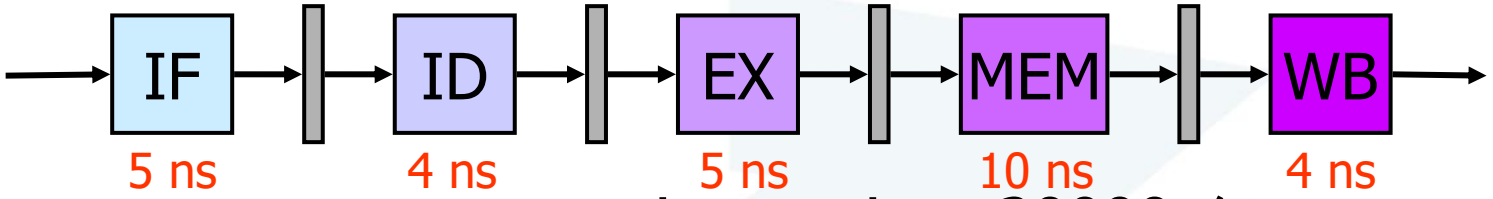


عندها يتحدد التأخير الكلي بزمن تأخير المرحلة الأبطأ



L(I1) = L(I2) = L(I3) = L(I4) = 50ns

تأخير وخرج توازي خط الأنابيب
Throughput and Latency



كم يستغرق تنفيذ 20000 instructions.

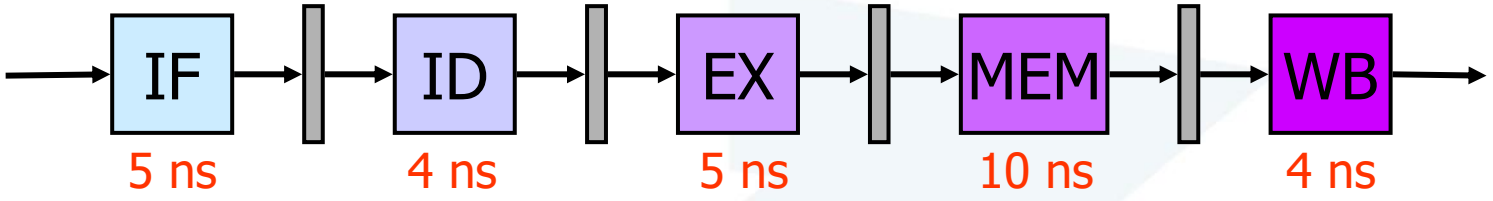
في الحالة المبينة بإهمال الأزمان الانتقالية مثل التفريع Branches وأخطاء النفاذ إلى الذاكرة المخبأية cache misses , والأخطار hazards. يكون الزمن:

$$ExecTime_{pipe} = 20000 \times 10ns = 200000ns = 200\mu s$$

بينما يستغرق نفس النموذج دون التوازي الأنبوبي:

$$ExecTime_{non-pipe} = 20000 \times 28ns = 560000ns = 560\mu s$$

تأخير وخرج توازي خط الأنابيب
Throughput and Latency



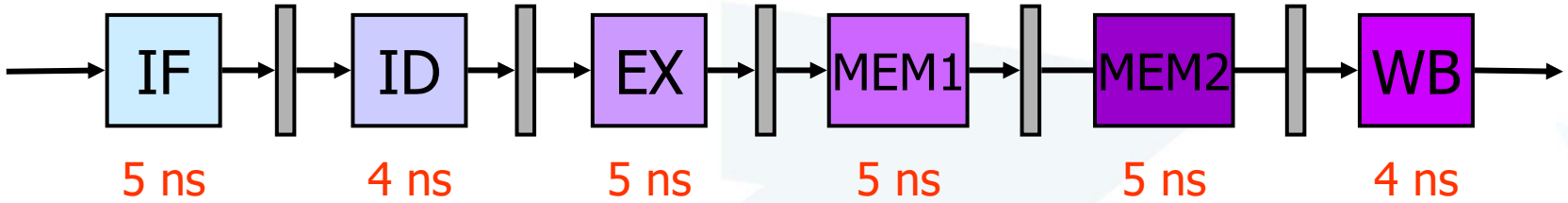
نسبة التسريع هي:

$$Speedup_{pipe} = \frac{ExecTime_{non-pipe}}{ExecTime_{pipe}} = \frac{560\mu s}{200\mu s} = 2.8$$

والسؤال هو كيف نستطيع تحسين التصميم الأنبوبي؟

نحتاج إلى تقليل **الاختلاف** في طول المراحل الأنبوبية لنزيد نسبة التسريع

Pipeline Throughput and Latency



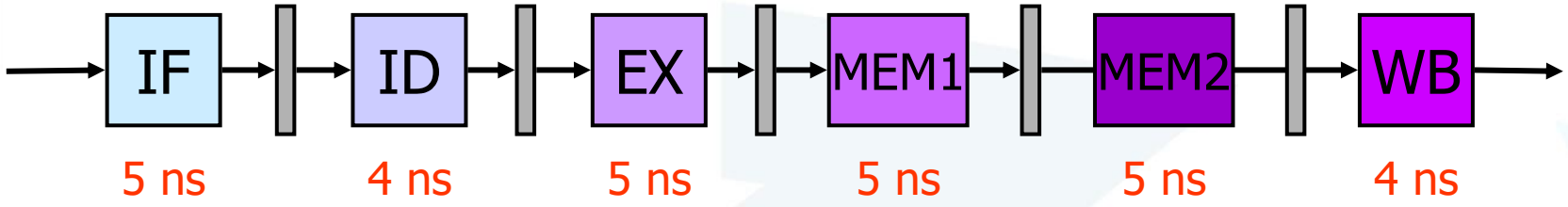
باضافة مرحلة أنبوبية جديدة مع المحافظة على زمن تنفيذ التعليمة الواحدة نقلل تأخير نفاذ المرحلة الأعظمي إلى النصف.

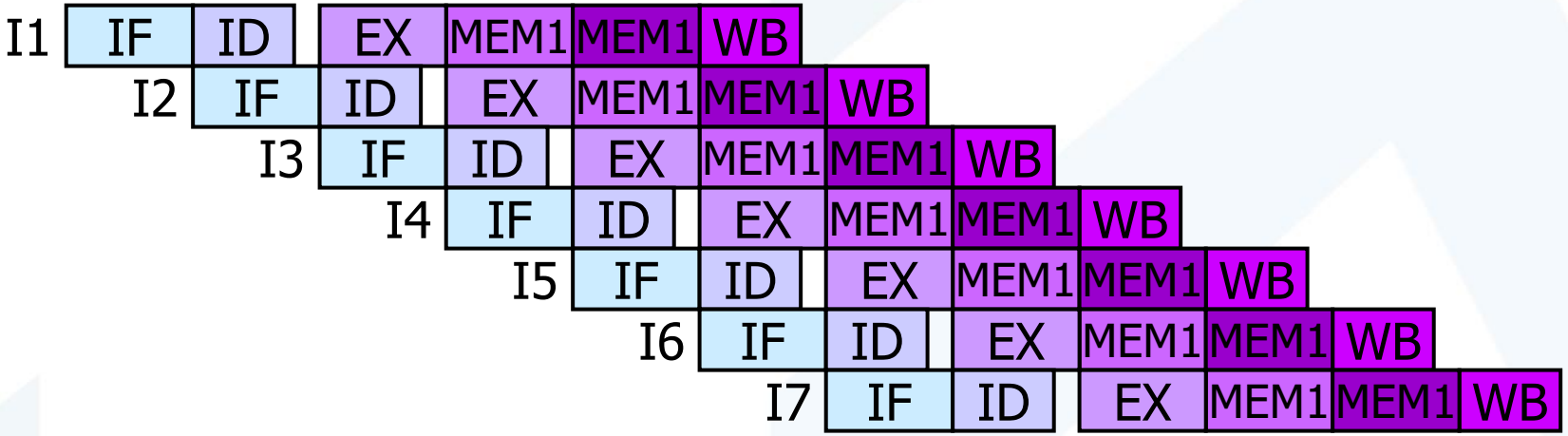
$$T = 1instr / \max(lat(IF), lat(ID), lat(EX), lat(MEM1), lat(MEM2), lat(WB))$$
$$= 1instr / \max(5ns, 4ns, 5ns, 5ns, 5ns, 4ns)$$
$$= 1instr / 5ns$$

تأخير تنفيذ التعليمة الواحدة انطلاقاً من تأخير نفاذ المرحلة هو:

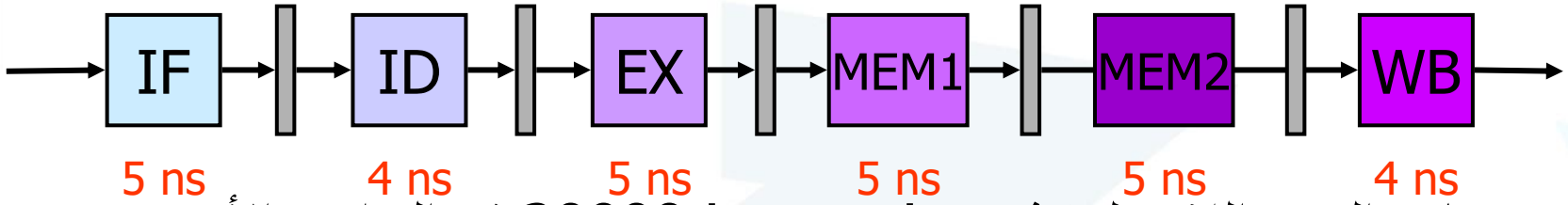
$$L = 6 \times 5ns = 30ns$$

Pipeline Throughput and Latency





Pipeline Throughput and Latency



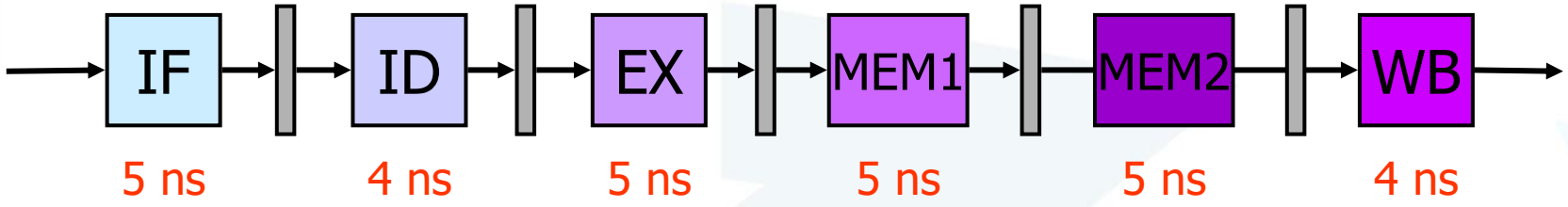
ماهو الزمن اللازم لتنفيذ 20000 instructions في التوازي الأنبوبي المبين أعلاه؟ (أهمل الضياع bubbles الناجم عن branches, cache misses, ..إلخ)

$$ExecTime_{pipe} = 20000 \times 5ns = 100000ns = 100\mu s$$

نسبة التسريع التي نحصل عليها هي:

$$Speedup_{pipe} = \frac{ExecTime_{non-pipe}}{ExecTime_{pipe}} = \frac{560\mu s}{100\mu s} = 5.6$$

Pipeline Throughput and Latency



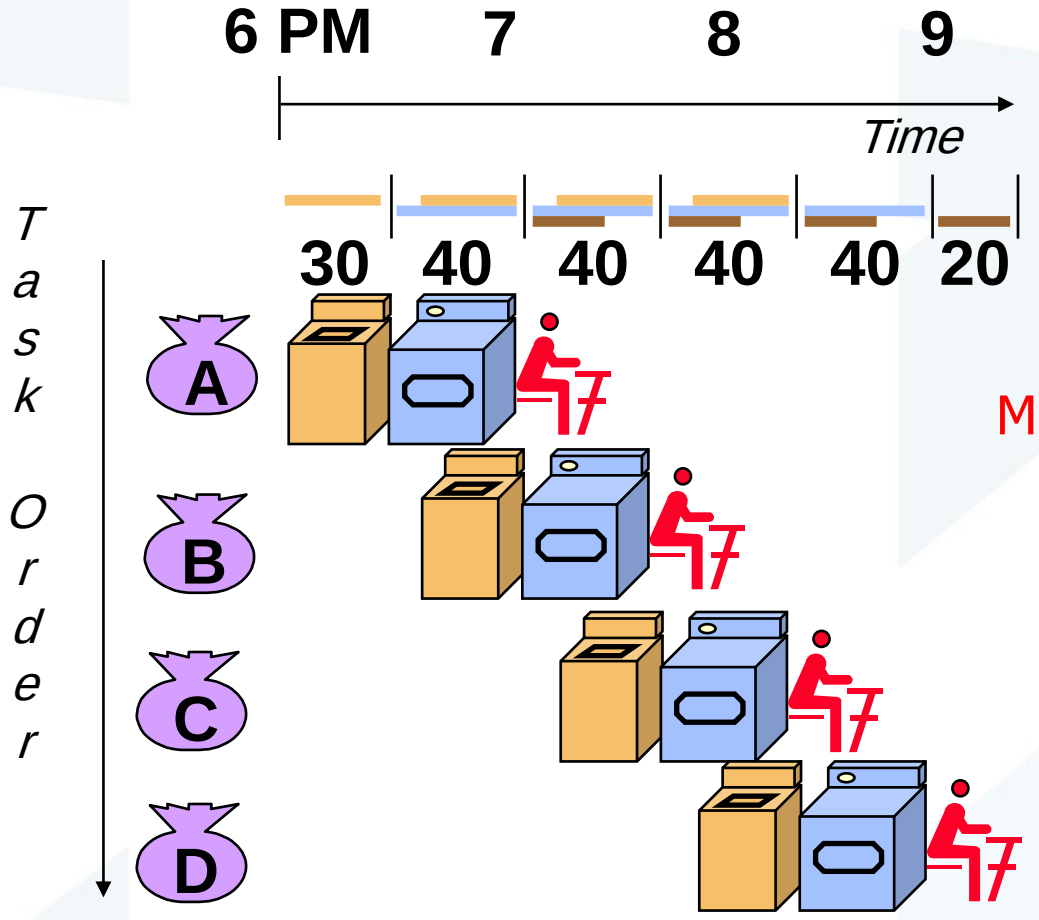
نستنتج مما سبق:

١-ضرورة موازنة التاخير في كل مرحلة من مراحل التوازي الأنبوبي

٢-تأخيرالنفاذ الأنبوبي Throughput هو 1/max(delay).

٣-التأخير Latency هو N×max(delay)، حيث
N هي عدد مراحل التوازي الأنبوبي.

Pipelining Lessons



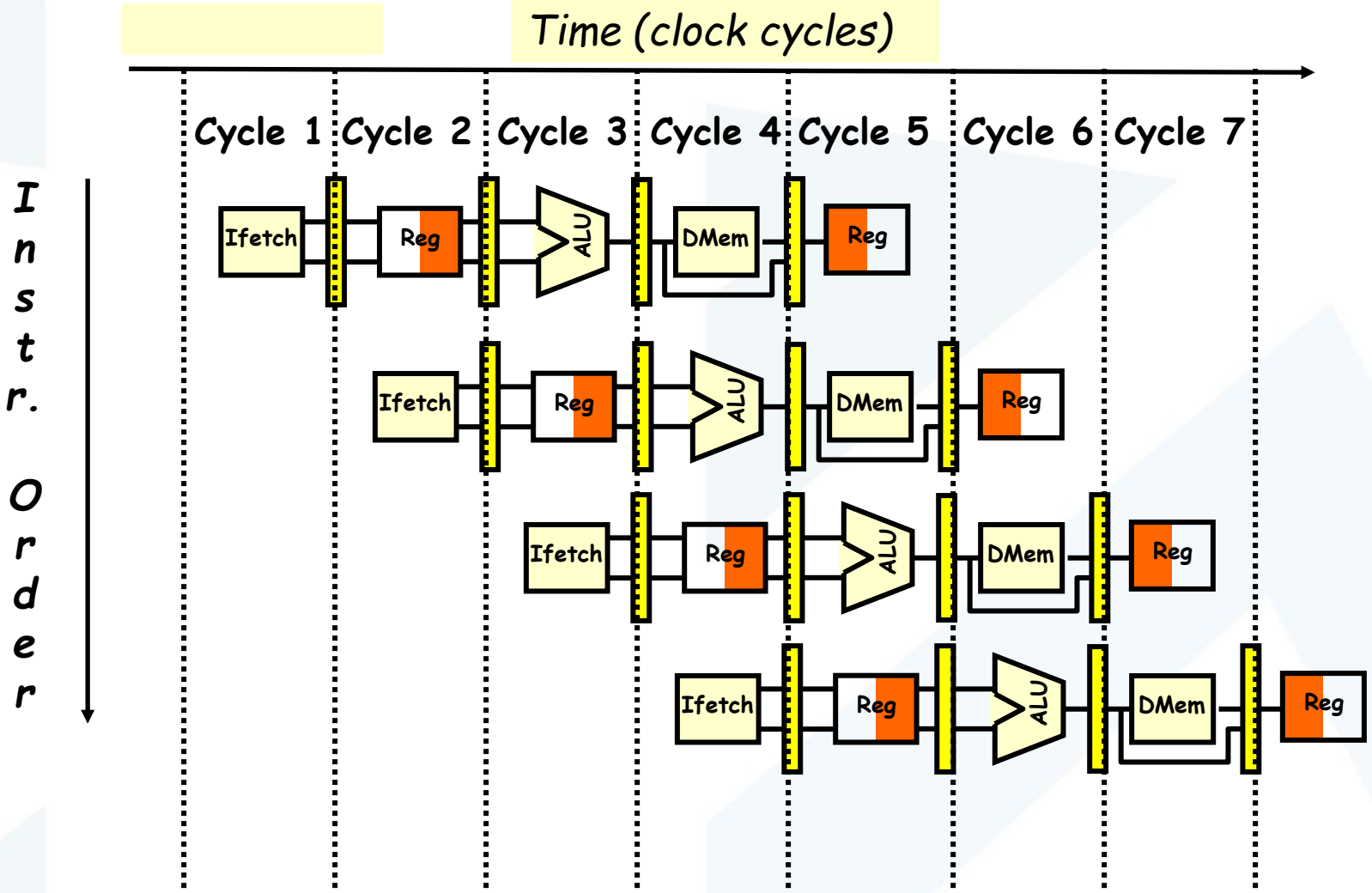
- التوازي الأنبوبي لا يقدم نتائج مناسبة للتأخير latency لمهمة واحدة ولكن يقدم تحسين للتأخير المرحلي throughput للحمل الكلي
- يتم ضبط إيقاع التوازي الأنبوبي وفقاً للمرحلة الأنبوبية الأبطأ
- تعالج المهمات المتعددة Multiple tasks بنفس الوقت simultaneously
- التسريع الكلي يتناسب مع عدد مراحل التوازي الأنبوبي
- الطول غير المتوازن للمراحل الأنبوبية يقلل نسبة زيادة السرعة
- تقلل أزمان الملء "fill" والتفريغ "drain" and من نسبة التسريع.

التوازي الأنبوبي في الحاسوب
Computer Pipelines

- تنفيذ مليارات التعليمات وبالتالي فإن المخرجات throughput المرحلية تقع في مركز الاهتمام.
- السمات المرغوبة:
  - جميع التعليمات بنفس الطول.
  - توضع المسجلات في نفس الموقع في هيكلية التعليمة instruction format.
  - التعامل مع حدود الذاكرة بعمليتي التحميل والتخزين فقط memory operands only in loads or stores

Visualizing Pipelining

Figure 3.3, Page 133 , CA:AQA 2e



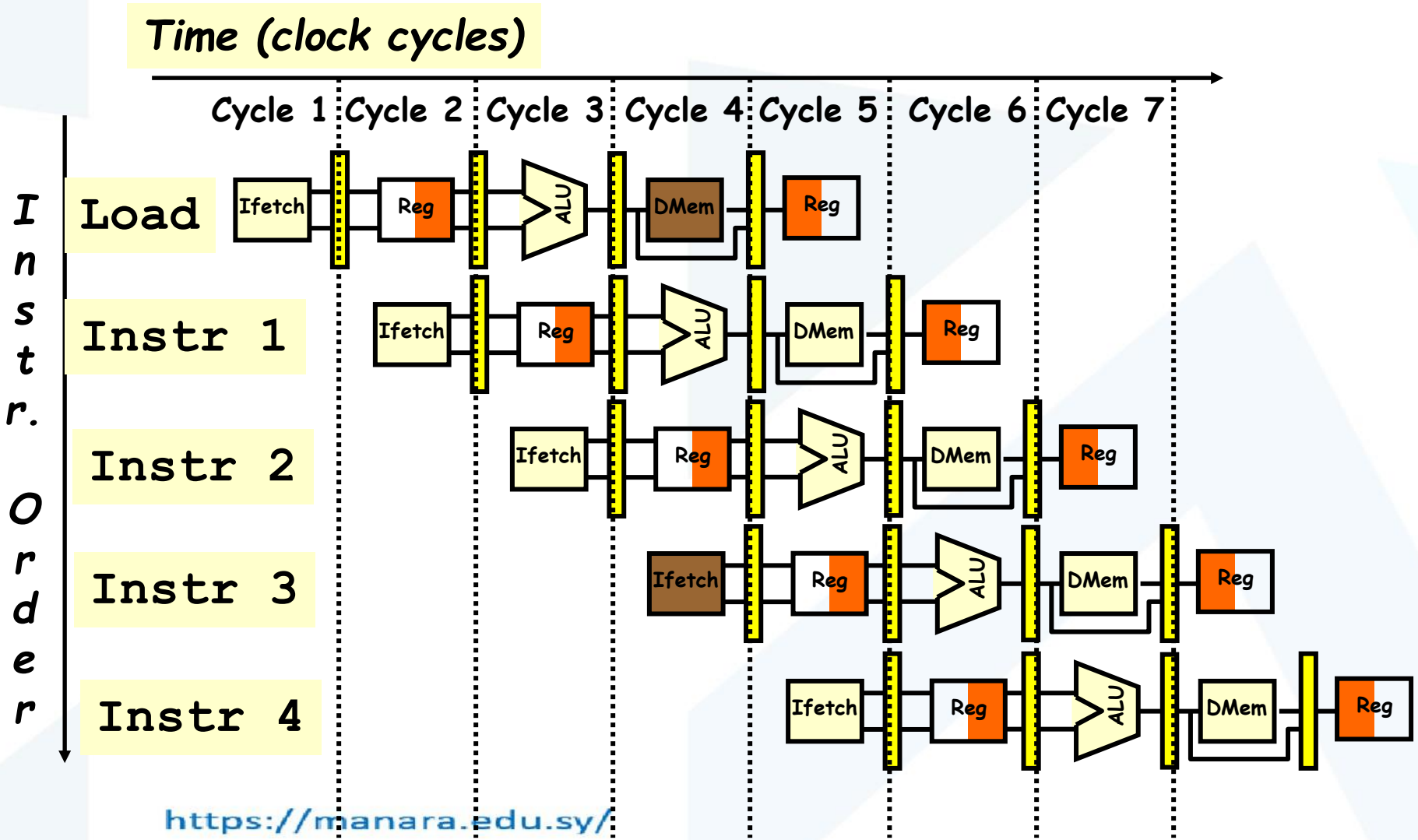
https://manara.edu.sy/

مشاكل التوازي الأنبوبي في المعالجات

- حدود التوازي الأنبوبي: تقوم الأخطار Hazards بمنع تنفيذ التعليمة التالية خلال دورة الساعة التي يتعرض فيها للخطر.
- أخطار بنيوية Structural hazards: لا يستطيع البناء المادي Hardware أن يدعم تركيبات التعليمات التي تتعرض للخطر البنيوي (single person to fold and put clothes away)
- أخطار البيانات Data hazards: اعتماد التعليمة على نتائج من التعليمات السابقة لا تزال قيد المعالجة الأنبوبية. (missing sock)
- أخطار تحكم Control hazards : سببها الاختلاف الزمني بين جلب التعليمة والقرار بإجراء تغيير السياق (القفز والتفريع)

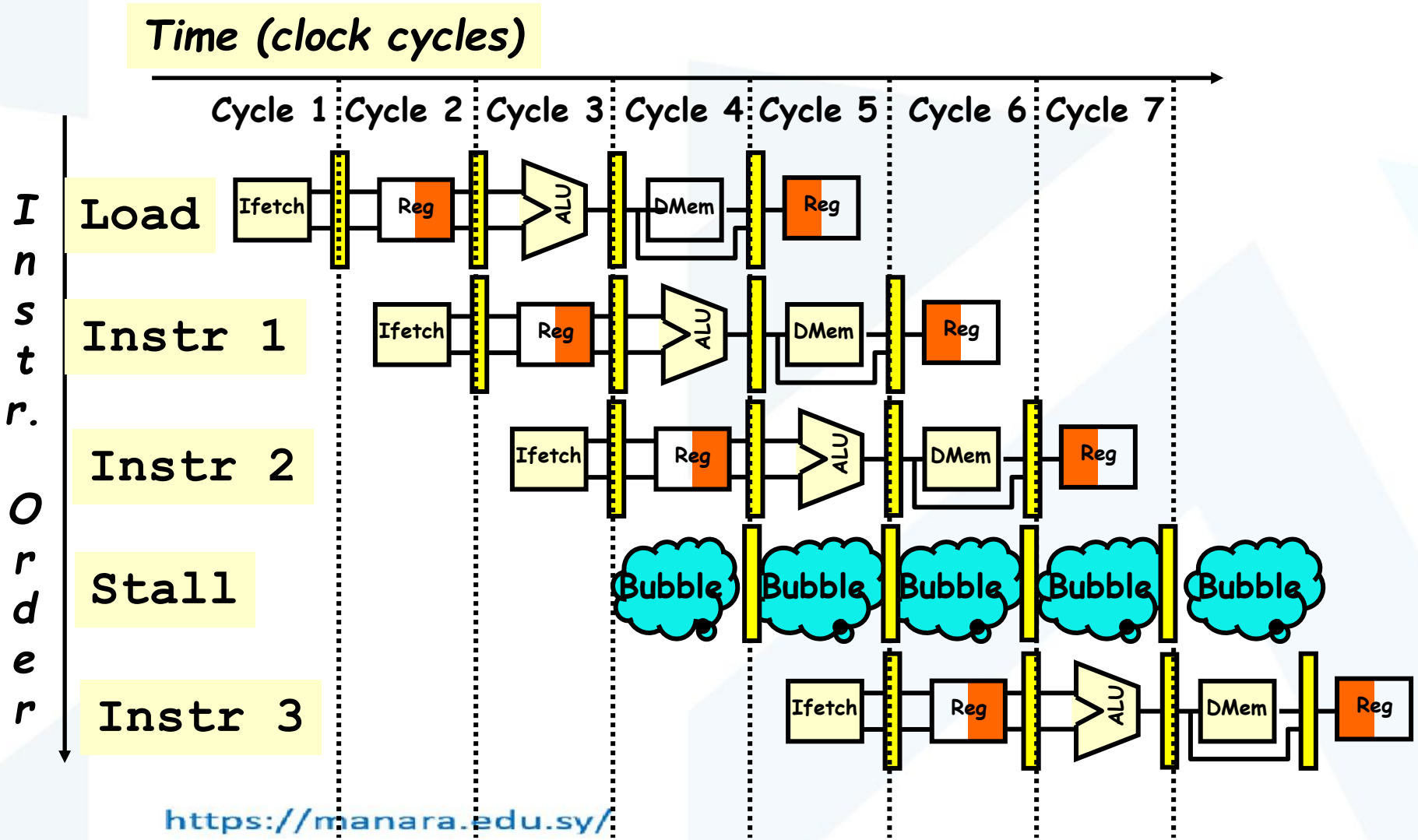
الأخطار البنيوية لذاكرة وحيدة البوابة

One Memory Port/Structural Hazards



الأخطار البنيوية لذاكرة وحيدة البوابة

One Memory Port/Structural Hazards



خطر البيانات على R1

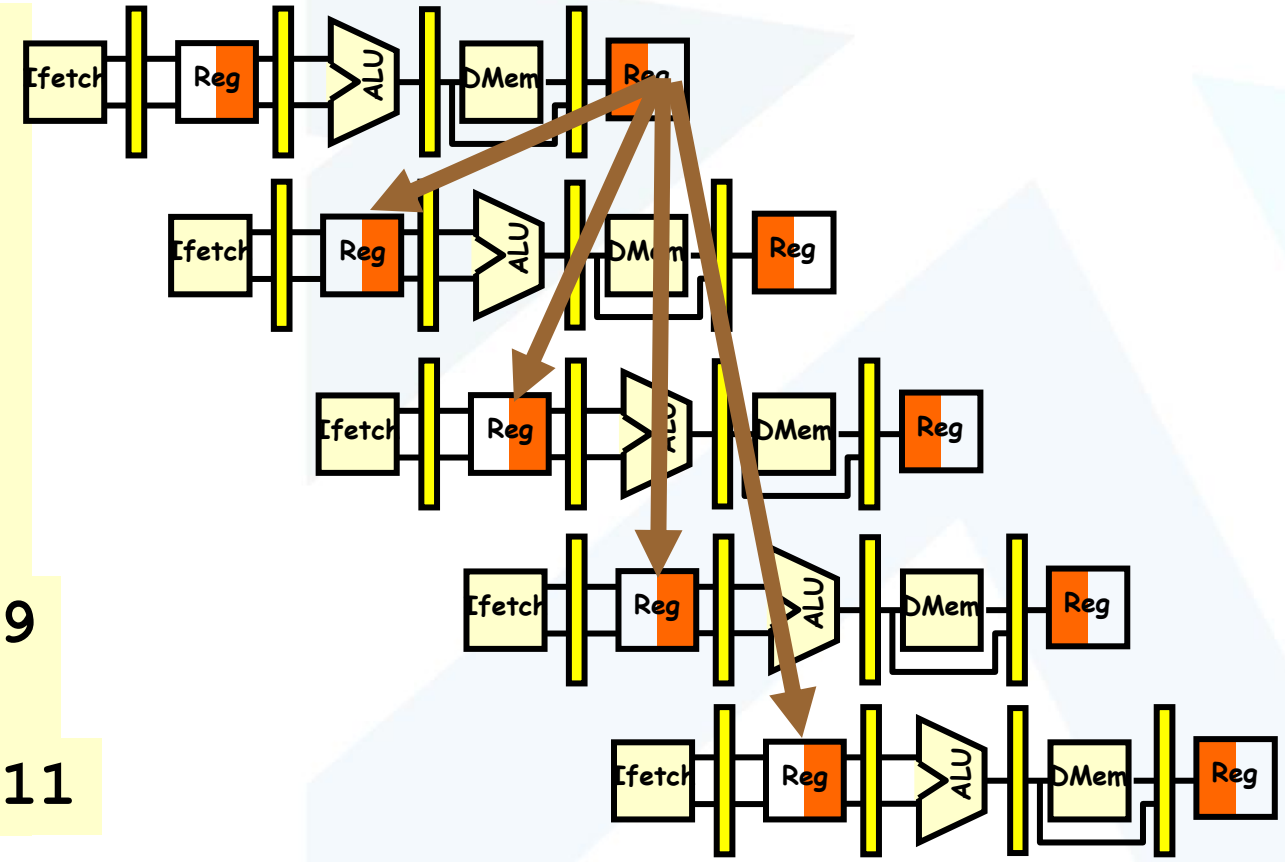
Data Hazard on R1

Time (clock cycles)

IF ID/RF EX MEM WB

Instr. Order

```
add r1,r2,r3
sub r4,r1,r3
and r6,r1,r7
or  r8,r1,r9
xor r10,r1,r11
```



https://manara.edu.sy/

أخطار البيانات الثلاثة الأساسية
Three Generic Data Hazards

- قراءة بعد كتابة Read After Write (RAW)

تحاول التعليمة $Instr_J$ أن تقرأ الحد قبل أن تقوم $Instr_I$ بكتابته.

```
I: add r1,r2,r3
J: sub r4,r1,r3
```

- سببها التبعية "Dependence" (in compiler) ويظهر هذا الخطر بسبب ضعف التواصل بين التعليمات.

أخطار البيانات الثلاثة الأساسية
Three Generic Data Hazards

- الكتابة بعد القراءة Write After Read (WAR)

$Instr_J$ تكتب الحد *before* قبل أن تقرأه التعليمة $Instr_I$.

```
I: sub r4,r1,r3
J: add r1,r2,r3
K: mul r6,r1,r7
```

- ويسمى من قبل المهتمين بالمترجمات بـالتبعية العكسية "anti-dependence".

وينجم عن إعادة استخدام الاسم "r1".

أخطار البيانات الثلاثة الأساسية
Three Generic Data Hazards

- الكتابة بعد الكتابة Write After Write (WAW)

$Instr_J$ تكتب الحد *before* قبل أن تكتبه $Instr_I$.

```
I: sub r1,r4,r3
J: add r1,r2,r3
K: mul r6,r1,r7
```

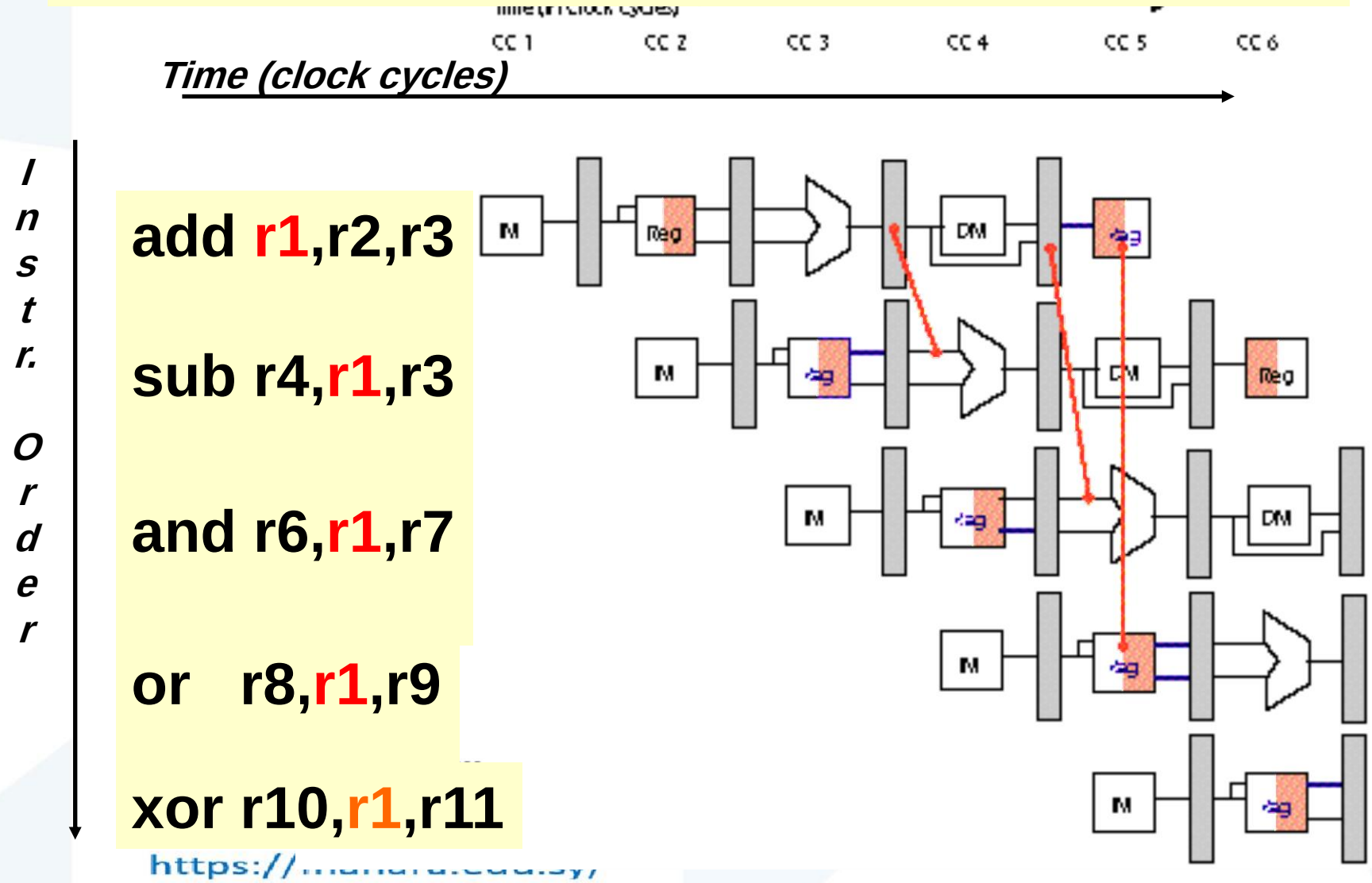
- يطلق عليه المهتمون بالمترجمات تبعية الخرج "output dependence".

وينجم أيضاً إعادة استخدام اسم "r1".

أخطار البيانات الثلاثة الأساسية
Three Generic Data Hazards

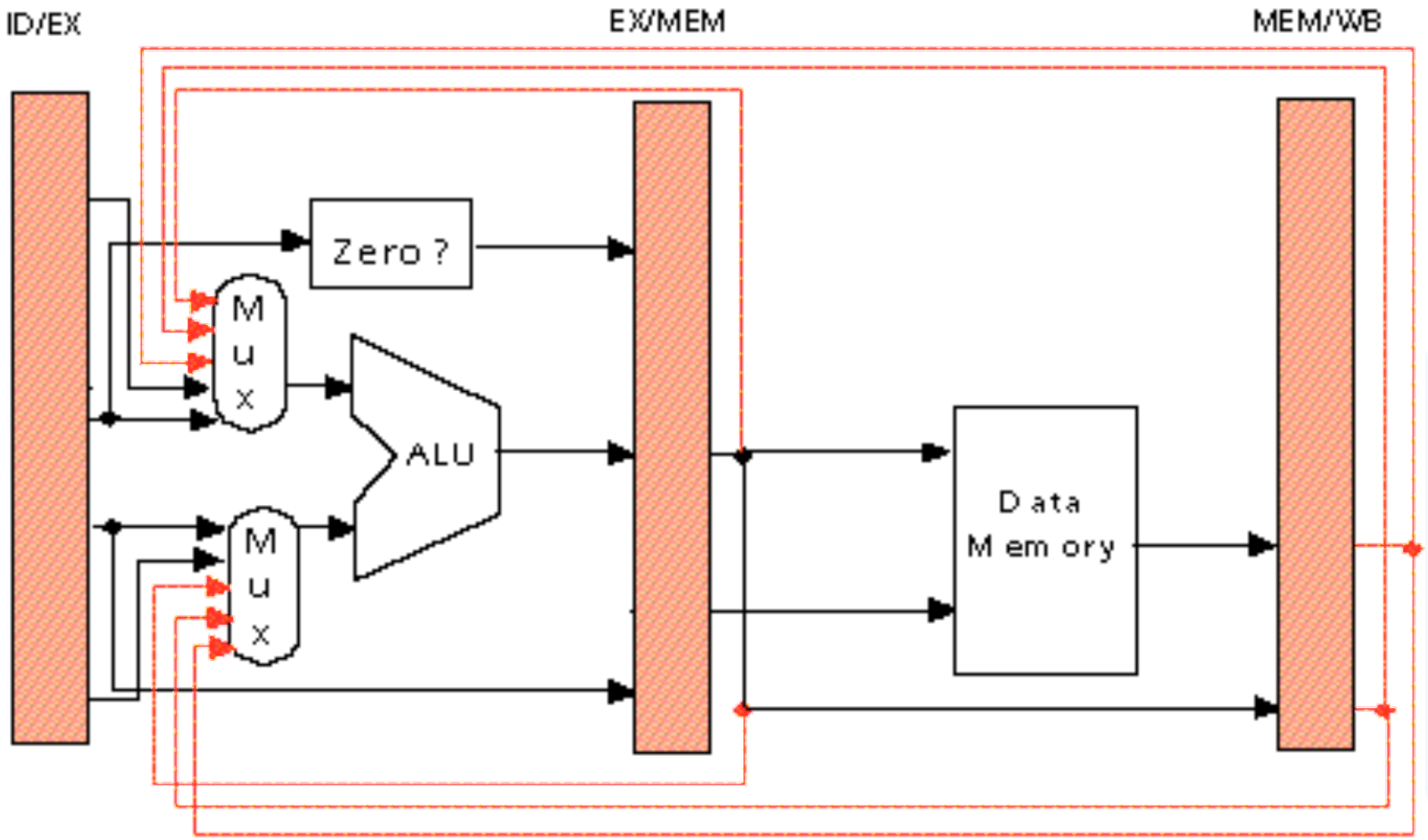
- لايمكن أن يحصل WAW في MIPSبـ 5 مراحل أنبوبية بسبب أن:
  - جميع التعليمات تأخذ 5 مراحل و
  - والكتابة دوماً في المرحلة 5.

تمرير البيانات إلى المراحل التالية لتجنب الخطر
Forwarding to Avoid Data Hazard



تغيير البنية المادية للتمرير إلى الأمام

HW Change for Forwarding



https://manara.edu.sy/

خطر البيانات رغم التمرير إلى الأمام

Time (clock cycles)

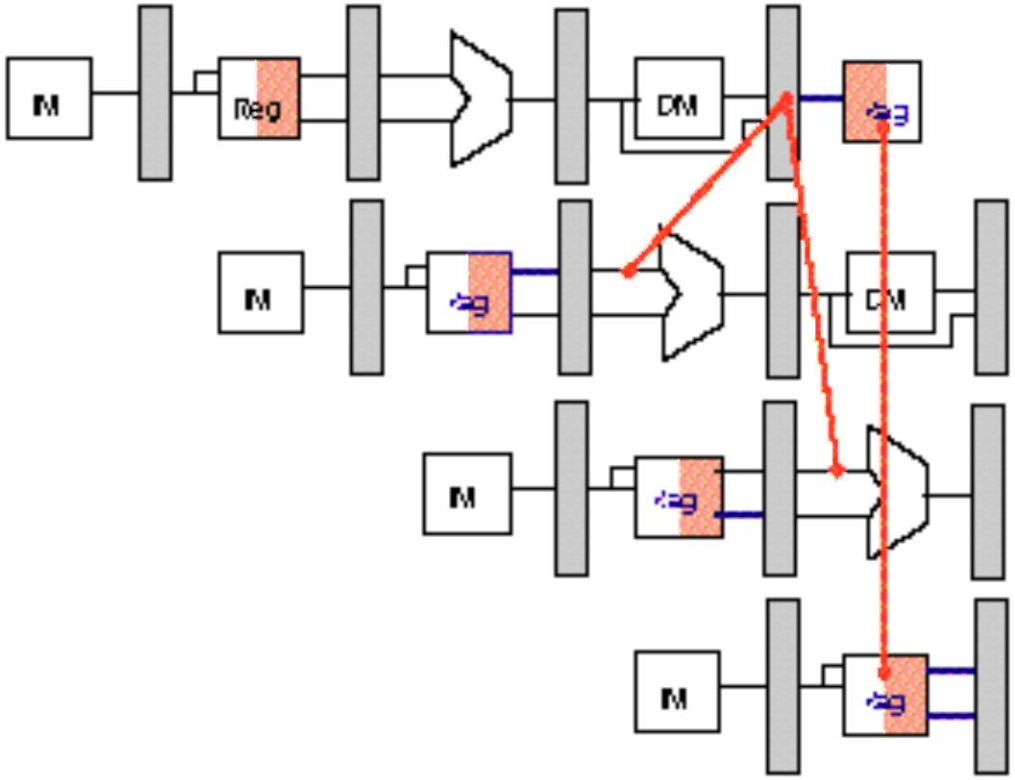
Instr. Order

lw r1, 0(r2)

sub r4,r1,r6

and r6,r1,r7

or r8,r1,r9



خطر البيانات رغم التمرير إلى الأمام

Time (clock cycles)

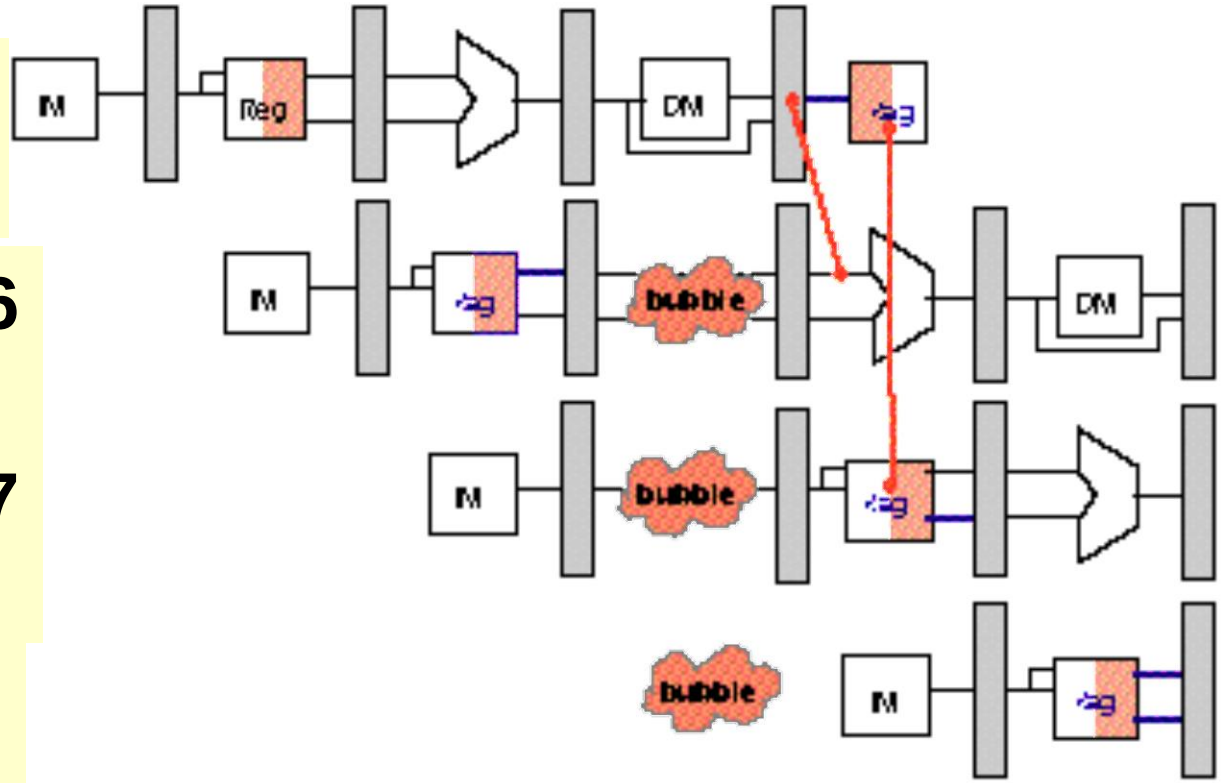
Instr. Order

lw r1, 0(r2)

sub r4,r1,r6

and r6,r1,r7

or r8,r1,r9



الجدولة البرمجية لتجنب أخطار التحميل

إذا حاولنا كتابة شيفرة سريعة للمعادلات:

a = b + c;

d = e – f;

بفرض أن **a, b, c, d ,e, and f** في الذاكرة.

Slow code:

LW	**Rb,b**
LW	**Rc,c**
ADD	**Ra,Rb,Rc**
SW	**a,Ra**
LW	**Re,e**
LW	**Rf,f**
SUB	**Rd,Re,Rf**
SW	**d,Rd**

Fast code:

LW	Rb,b
LW	Rc,c
LW	Re,e
ADD	Ra,Rb,Rc
LW	Rf,f
SW	a,Ra
SUB	Rd,Re,Rf
SW	d,Rd

https://manara.edu.sy/

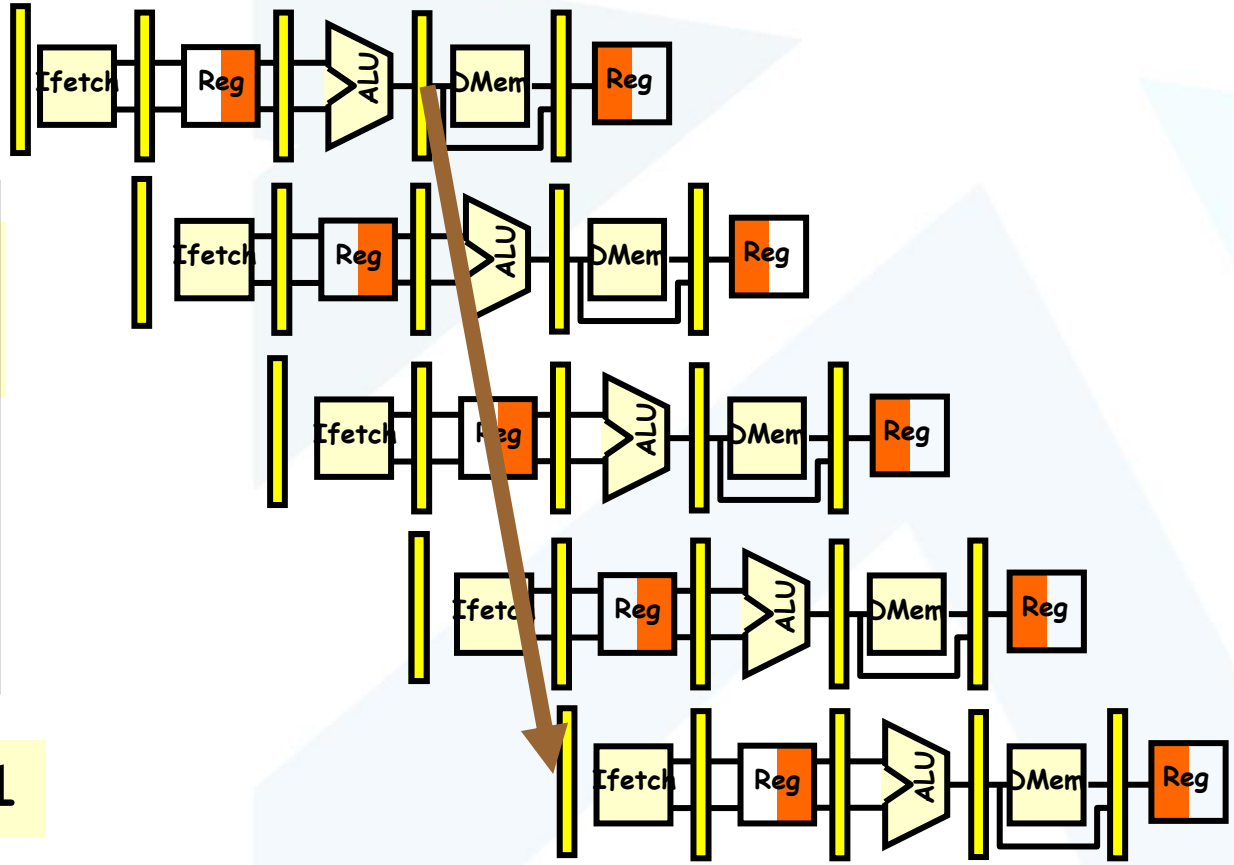
لماذا تكون الشيفرة أسرع

Cycle	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
LW Rb, b	IF	ID	EX	MEM	WB										
LW Rc, c		IF	ID	EX	MEM	WB									
ADD Ra, Rb, Rc			IF	ID	**stall**	EX	MEM	WB							
SW a, Ra				IF	**stall**	ID	EX	MEM	WB						
LW Re, e					**stall**	IF	ID	EX	MEM	WB					
LW Rf, f							IF	ID	EX	MEM	WB				
SUB Rd, Re, Rf								IF	ID	**stall**	EX	MEM	WB		
SW d, Rd									IF	**stall**	ID	EX	MEM	WB	

Cycle	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
LW Rb, b	IF	ID	EX	MEM	WB										
LW Rc, c		IF	ID	EX	MEM	WB									
LW Re, e			IF	ID	EX	MEM	WB								
ADD Ra, Rb, Rc				IF	ID	EX	MEM	WB							
LW Rf, f					IF	ID	EX	MEM	WB						
SW a, Ra						IF	ID	EX	MEM	WB					
SUB Rd, Re, Rf							IF	ID	EX	MEM	WB				
SW d, Rd								IF	ID	EX	MEM	WB			

الخطر التحكمي Control Hazard عند التفريع Branches
تعطيل ثلاثي المراحل Three Stage Stall

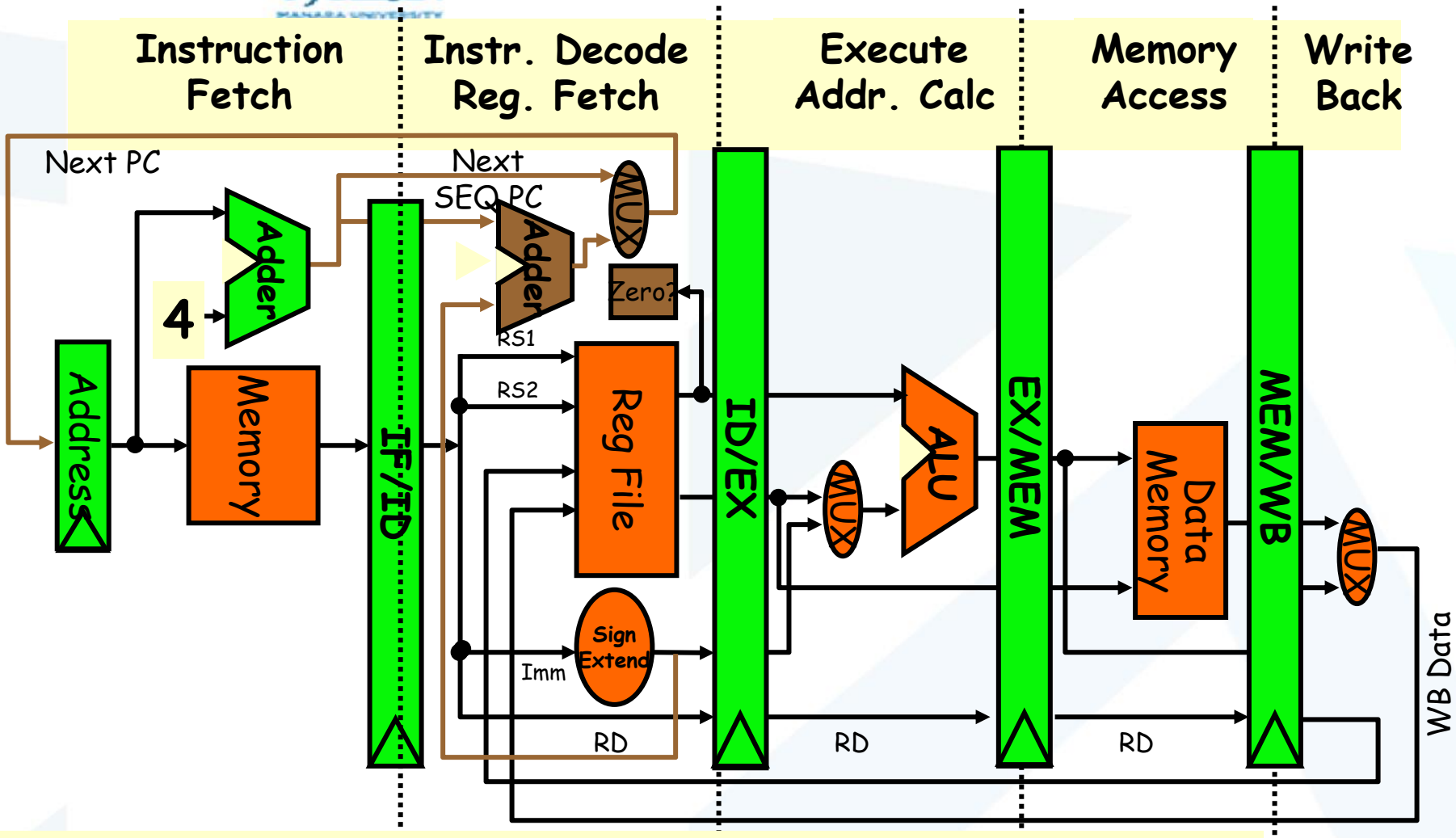
```
10: beq r1,r3,36

14: and r2,r3,r5

18: or  r6,r1,r7

22: add r8,r1,r9

36: xor r10,r1,r11
```



مثال: تأثير تعطيل التفرع Branch Stall Impact

- إذا كان هنالك 30% من التعليمات تعليمات تفرع branch فإن تأثير تعطيل 3 cycles يكون كبيراً.
- الحل على مرحلتين Two part solution:
  - التحديد الفوري لاعتماد التفريع أو تخطيه و
  - حساب عنوان التفريع في حال اعتماده بشكل مسبق.
- MIPS يختبر التفريع إذا كان register = 0 or ≠ 0
- الحل المقدم من MIPS:
  - نقل اختبار الصفر إلى مرحلة ID/RF.
  - استخدام جامع لحساب PC الجديد في مرحلة ID/RF
  - 1 clock cycle جزاء تأخير التفريع بدلاً من 3.

ممر بيانات التوازي الأنبوبي لـ MIPS



- تحكم ثابت بالبيانات Data stationary control

-فك ترميز محلي في طور التعليمة بدلاً من مرحلة التوازي الأنبوبي.

أشكال أخطار التفريع الأربع
Branch Hazard Alternatives

#1: التعطيل حتى وضوح اتجاه التفريع.

#2: التنبؤ بعدم اعتماد التفريع:

- تنفيذ التعليمات التالية على التوالي.
- حشر "Squash" تعليمات في المسار الأنبوبي إذا تم اعتماد التفريع
- ميزات التحديث المتأخر للحالة الأنبوبية.
- مايقرب من 47% من تفريعات الـ MIPS لا تعتمد.
- حساب PC+4 المسبق يمكننا من استخدامه في الحصول على التعليمة التالية

#3: اعتماد التفريع المتوقع

- يتم اعتماد مايقرب من 53% من تفريعات MIPS.
- إلا أننا لانكون قد حسبنا عنوان التفريع الهدف في MIPS.
  - وتبقى معالجات MIPS تعاني من جزاء تفريع يبلغ 1 cycle.
  - في المعالجات الأخرى : يكون هدف التفريع معروفاً قبل الخروج.

أشكال أخطار التفريع الأربع

Branch Hazard Alternatives

#4: التفريع المؤخر:

- تعريف التفريع بحيث يحدث بعد AFTER التعليمة التالية.

```
branch instruction
  sequential successor1
  sequential successor2
  ........
  sequential successorn
branch target if taken
```

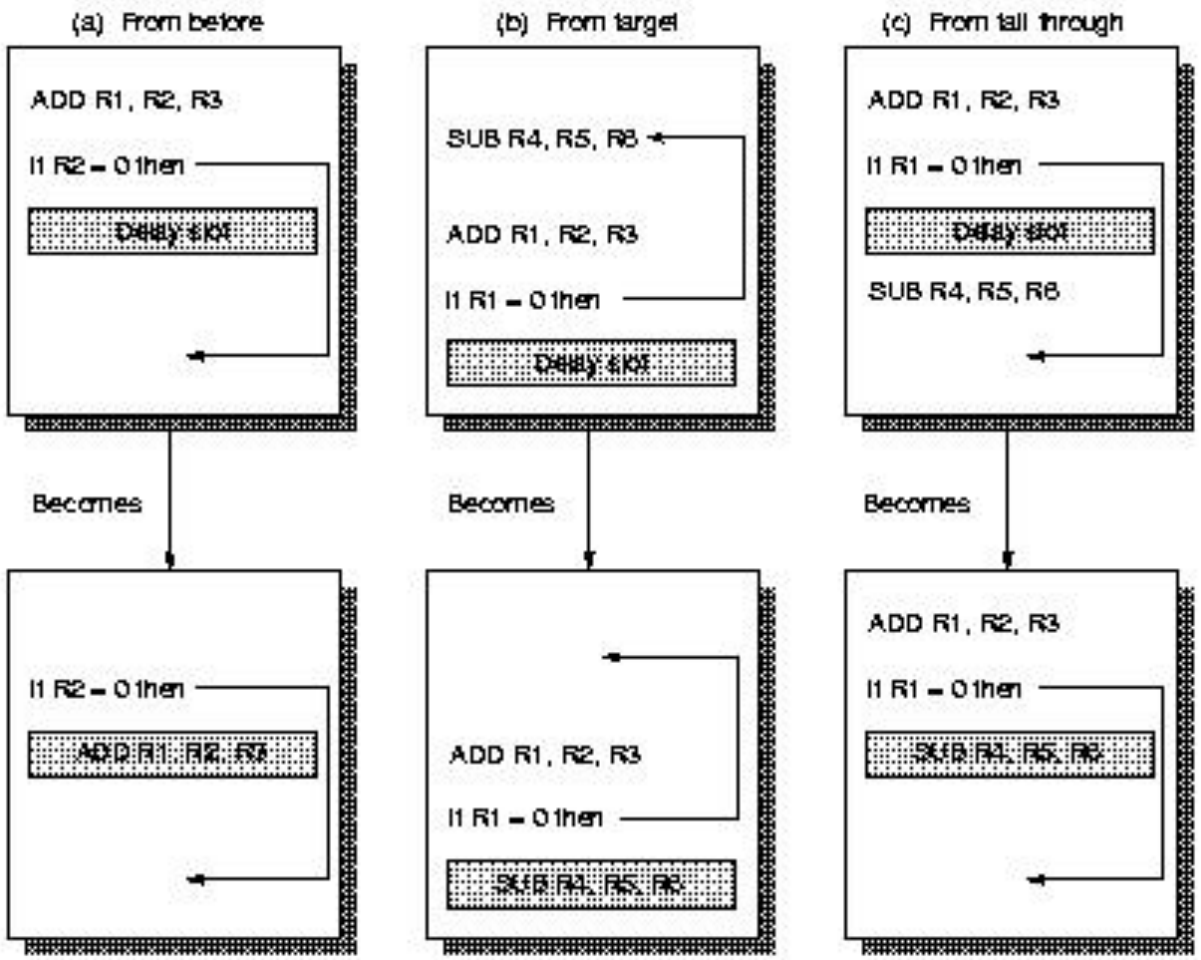
Branch delay of length *n*

- يسمح التأخير لشريحة زمنية واحدة باتخاذ القرار المناسب والتفريع للعنوان الهدف ب 5 مراحل أنبوبية.
- تستخدم معالجات MIPS المذكور أعلاه.

التفريع المؤخر (المؤجل)
Delayed Branch

- من أين سنؤمن التعليمات لملء شريحة التأخير الزمنية الناجمة عن التفريع؟
  - قبل تعليمة التفريع.
  - العنوان الهدف: له قيمة فقط عندما يعتمد التفريع.
  - الإسقاط عند التخلي عن التفريع: له قيمة فقط في حال عدم اعتماد التفريع
  - يسمح إلغاء التفريعات بملء عدة شرائح زمنية.to be filled
    - إلغاء التفريع ينفذ فقط التعليمات الموجودة في شريحة التأخير delay slot إذا كان اتجاه التنبؤ صحيحاً.

Fig. 3.28


(a) From before
ADD R1, R2, R3
I1 R2 = 0 then
Delay slot
Becomes
I1 R2 = 0 then
ADD R1, R2, R3
(b) From target
SUB R4, R5, R8
ADD R1, R2, R3
I1 R1 = 0 then
Delay slot
Becomes
ADD R1, R2, R3
I1 R1 = 0 then
SUB R4, R5, R8
(c) From fall through
ADD R1, R2, R3
I1 R1 = 0 then
Delay slot
SUB R4, R5, R8
Becomes
ADD R1, R2, R3
I1 R1 = 0 then
SUB R4, R5, R8

Delayed Branch التفريع المؤجل

- فعالية المترجم من أجل تأجيل شريحة واحدة تعتمد على:
  - ملء 60% من شرائح التأخير تقريباً
  - يدخل حيز التنفيذ مايقارب من 80% من التعليمات الموجودة في شرائح تأخير التفريع of
  - نستفيد من مايقارب (60% x 80%) 50% من الشرائح المملوءة.
- سلبيات التفريع المؤجل تتجلى في: الحاجة إلى من 7-8 مراحل أنبوبية، والإصدار لأكثر من تعليمة في كل دورة ساعة(تدرجية فائقة superscalar)