جامعة المنارة الخاصة
كلـــــية الهندســـــة
هندسة ميـــكاترونيك

المعالجات الصغرية ولغة التجميع

المحاضرة الأولى

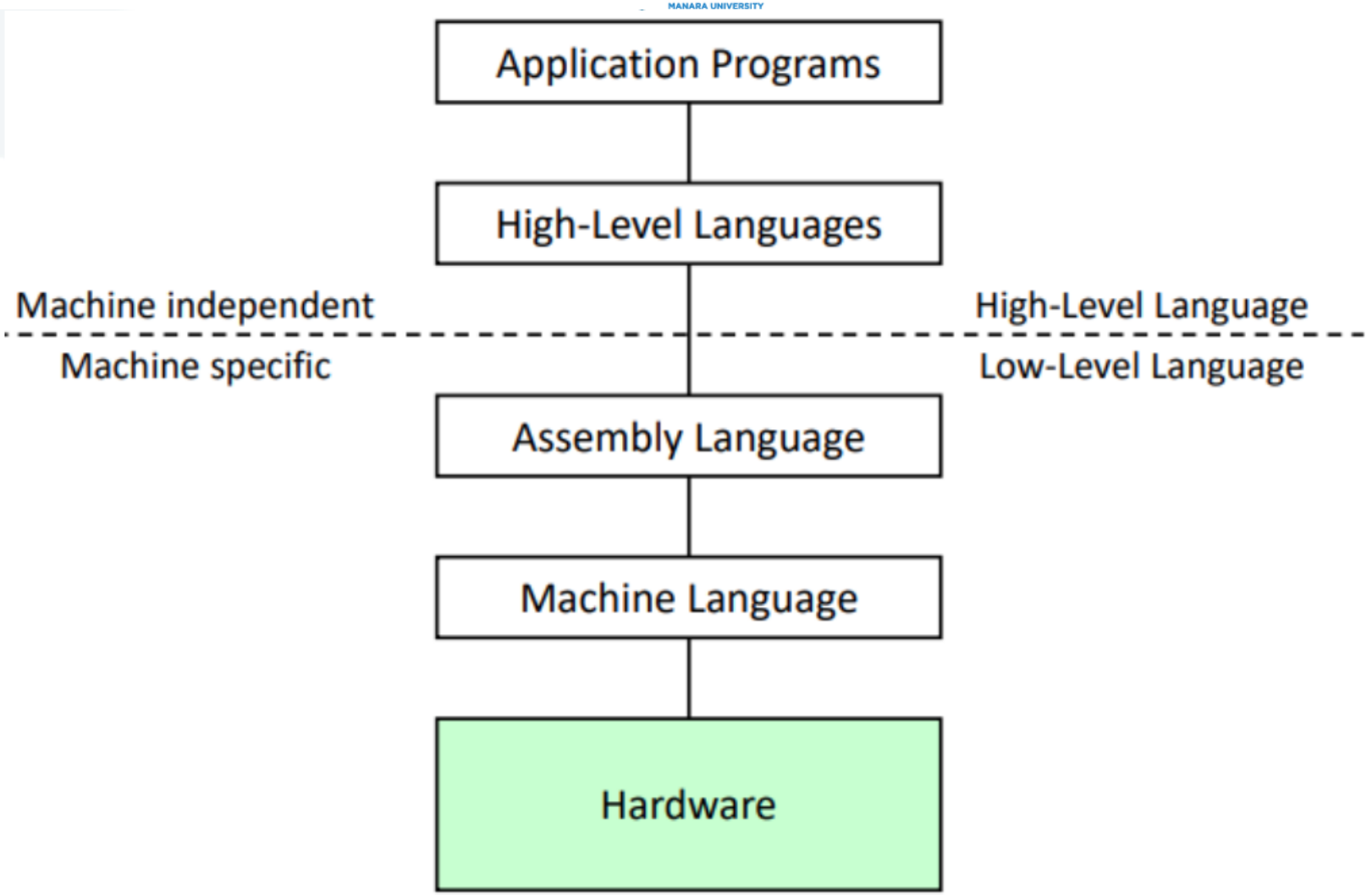
مدرس المقرر
د. بسام حسن

2025_2026

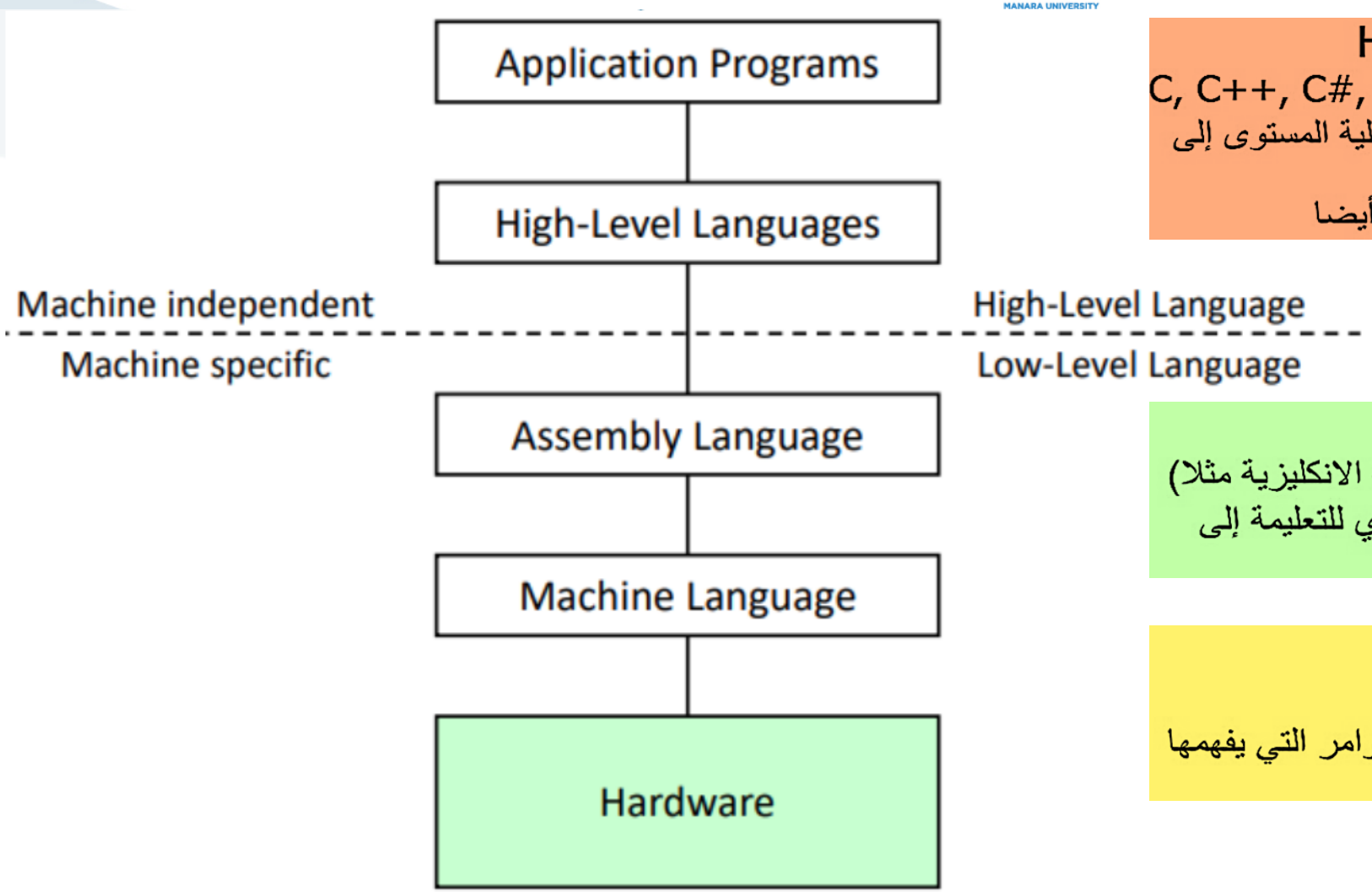
مفردات المحاضرة الأولى :

- التسلسل الهرمي للغات البرمجة.
- المترجم Compilerوالمجمّع Assembler
- معمارية مجموعة التعليمات ISA
- ISA و نموذج MIPS
- تمثيل البيانات وأنظمة العد

التسلسل الهرمي للغات البرمجة



التسلسل الهرمي للغات البرمجة



لغة عالية المستوى High-Level Languages

- لغة تتألف من كلمات وصيغ جبرية, مثال: لغة C, C++, C#, Java
- المترجم Compiler: برنامج يحول عبارات اللغة عالية المستوى إلى عبارات بلغة التجميع
- ملاحظة: تقوم المترجمات عمليا بإنجاز عمل المجمع أيضا

لغة التجميع Assembly Language

- هي تمثيل رمزي لتعليمات الآلة (تمثيل رمزي باللغة الانكليزية مثلا)
- المجمع Assembler: برنامج يحول التمثيل الرمزي للتعليمة إلى صيغتها الثنائية

لغة الآلة Machine Language

- هي التمثيل الثنائي لتعليمات الآلة
- تعليمات الآلة Machine Instructions: هي الأوامر التي يفهمها الكيان الصلب للحاسب وينفذها.

ترجمة لغات البرمجة عالية المستوى إلى لغة الآلة

Program (C Language):

```
swap(int v[], int k) {
   int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

Compiler

MIPS Assembly Language:

```
sll $2,$5, 2
add $2,$4,$2
lw  $15,0($2)
lw  $16,4($2)
sw  $16,0($2)
sw  $15,4($2)
jr  $31
```

Assembler

MIPS Machine Language:

```
00051080
00821020
8C620000
8CF20004
ACF20000
AC620004
03E00008
```

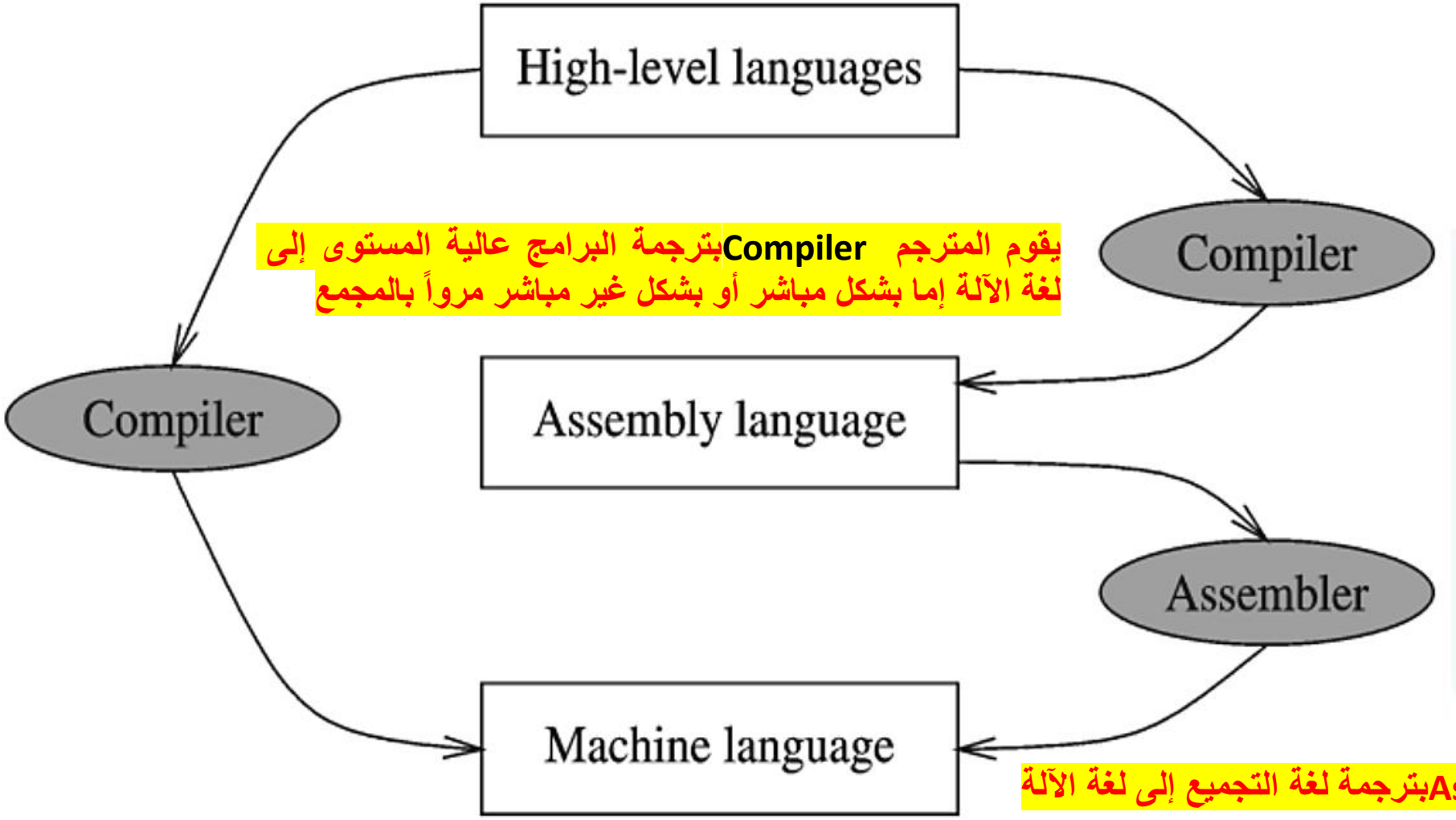
A statement in a high-level language is translated typically into several machine-level instructions

ملاحظة

التعليمات في لغة الآلة تتكون من كود ثنائي: 1و 0

كل تعليمة في لغة التجميع يقابلها تعليمة في لغة الآلة (تعليمة مقابل تعليمة)

المترجم و المجمع
Compiler and Assembler



يقوم المترجم Compilerبترجمة البرامج عالية المستوى إلى لغة الآلة إما بشكل مباشر أو بشكل غير مباشر مروراً بالمجمع

يقوم المجمع Assemblerبترجمة لغة التجميع إلى لغة الآلة

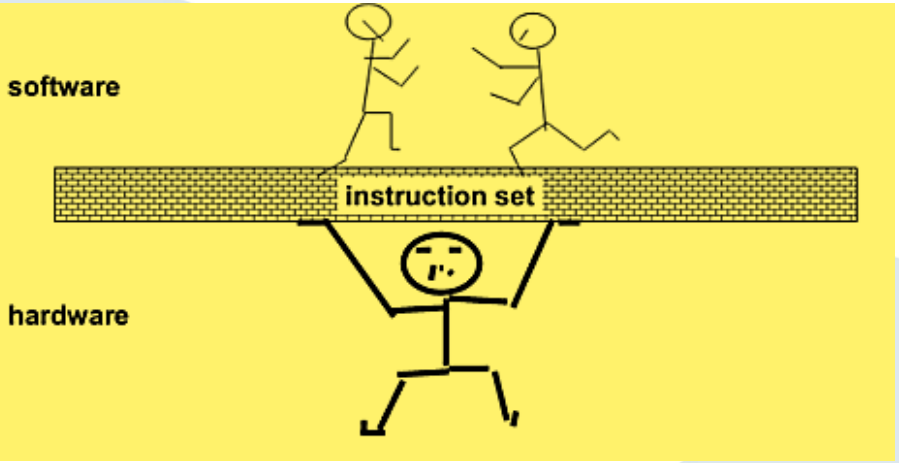
تعليمات ولغة الآلة

- لغة الآلة هي اللغة الأم للمعالج : يتم تنفيذها مباشرة بواسطة الدارات المكونة للمعالج.
- يطلق على كل أمر في البرنامج تعليمة (يوجه الكمبيوتر إلى ما يجب القيام به)
- تتعامل أجهزة الكمبيوتر مع البيانات الثنائية فقط، وبالتالي يجب أن تكون
- التعليمات بتنسيق ثنائي (0و 1).
- المجموعة المكونة من جميع التعليمات (في شكل ثنائي) تشكل لغة الآلة للكمبيوتر

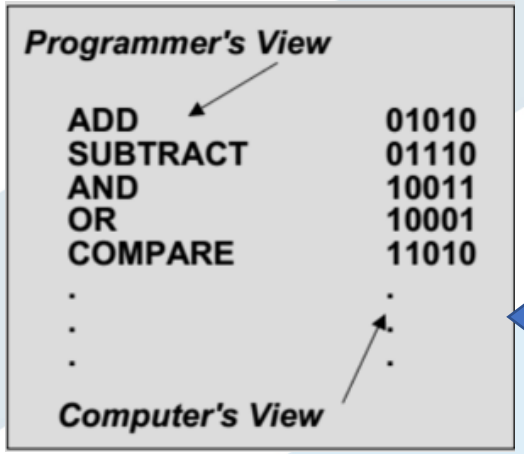
Instruction Set Architecture :ISA

معمارية مجموعة التعليمات

ISA يجعل البرامج تعمل على العتاد بشكل موحد مهما اختلف نوع أو تفاصيل العتاد.



- معمارية مجموعة التعليمات **ISA** :هي حلقة الوصل بين البرمجيات والعتاد في الحاسوب، أي أنها تحدد كيف تستطيع البرامج أن تتخاطب مع المكونات الفيزيائية للجهاز.
- **ISA** تعرف ما يمكن للبرنامج رؤيته من حالات (مثل سجلات المعالج والذاكرة) وتحدد شكل التعليمات ومعانيها (أي كيفية ترميز التعليمات ومعاني العمليات).
- هناك أمثلة مشهورة على ISAمثل IBM 360 وMIPS وRISC-V وx86
- البرنامج (سوفتوير) يعتمد على ISA كجسر للوصول إلى العتاد (هاردوير).
- "**instruction set**"هي مجموعة التعليمات التي يمكن للبرنامج كتابتها، بينما يقوم العتاد بتنفيذها فعليا

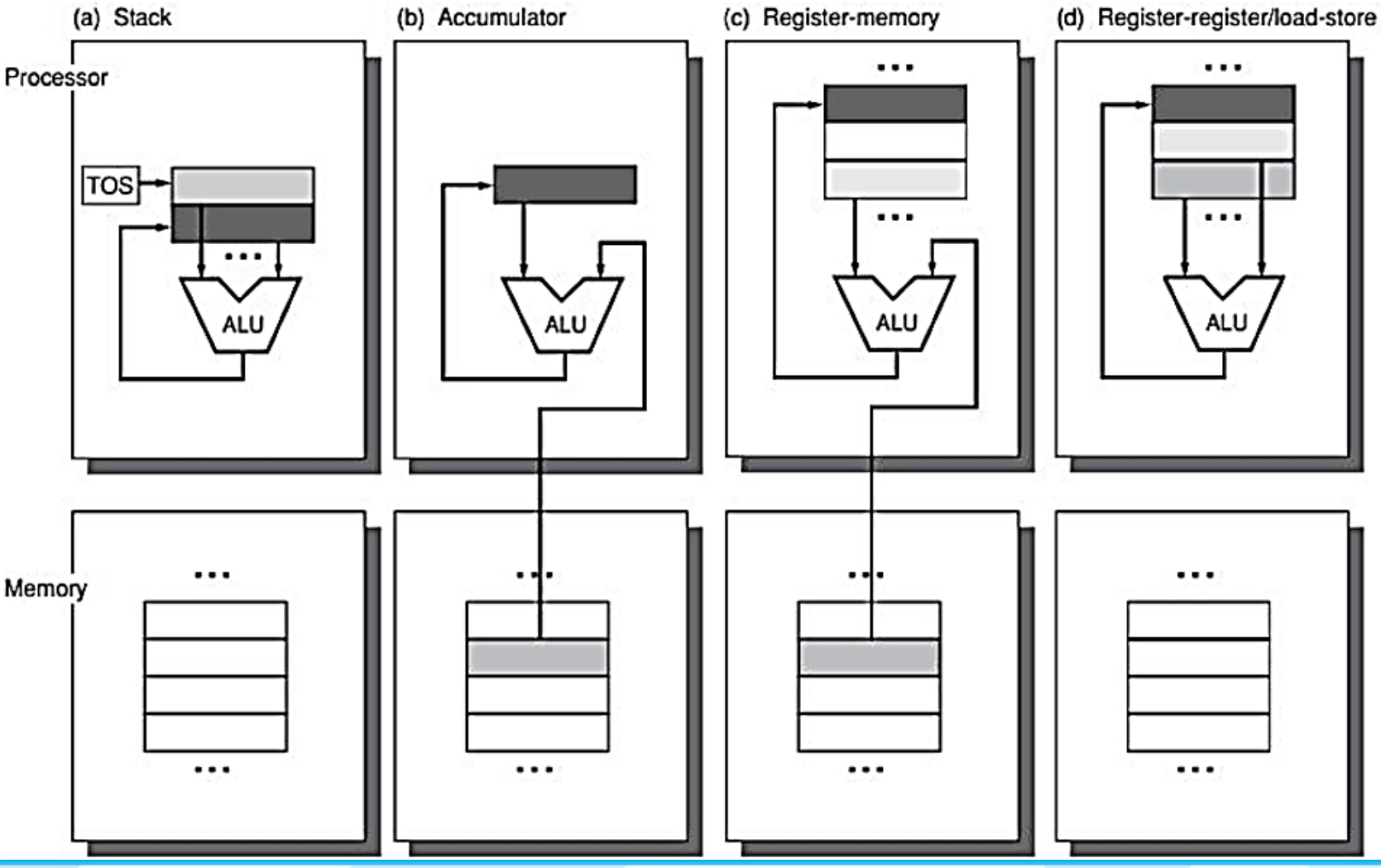


- يرى المبرمج التعليمات مثلا: ADD أو SUBTRACT أو AND،
- بينما يراها الحاسوب على شكل رموز ثنائية (مثل: 01010 أو 01100).

Instruction Set Architecture :ISA

معمارية مجموعة التعليمات

يوضح الشكل :طرق التعامل مع البيانات (المعاملات operand)
بين المعالج والذاكرة في تصميم الحواسيب، من خلال أربعة نماذج مشهورة لتعليمات معمارية المعالج:



(d) Register-Register (Load-Store)

- النموذج يعتمد أساساً على المسجلات داخل المعالج فقط في العمليات الحسابية
load-store architecture

- الإدخال والإخراج للعمليات يتم فقط بين المسجلات، أما نقل البيانات من وإلى الذاكرة فيتم بتعليمات منفصلة وهي:
للتحميل load أو **التخزين store**

- النموذج هذا مستخدم على نطاق واسع في معالجات RISCمثل **MIPS** و ARM

Advantages of General Purpose Register

فوائد مسجلات الأغراض العامة:

- كل المسجلات أسرع من الذاكرة.
- المسجلات أكثر كفاءة ليستخدمها المترجم من صيغ التخزين الداخلي الآخرى.
- عندما تتوضع المعاملات في مسجلات يخف ازدحام الذاكرة وتزداد سرعة البرنامج (لأن المسجلات اسرع من الذاكرة) إضافة إلى ذلك تتحسن كثافة الترميز أو العنونة (لأن المسجلات يمكن ترميزها بعدد أقل من البيتات على عكس مواقع الذاكرة.
- المسجلات تمكننا من التنفيذ المتوازي للتعليمات (مستويات متوازية):

عدد المعاملات وصيغ المعاملات بين المعماريات

كم عدد **المعاملات** **operand** في تعليمات ALU

- صيغة الثلاثة معاملات: تحتوي التعليمة على معامل نتيجة واحد ومعاملين مصدرين.

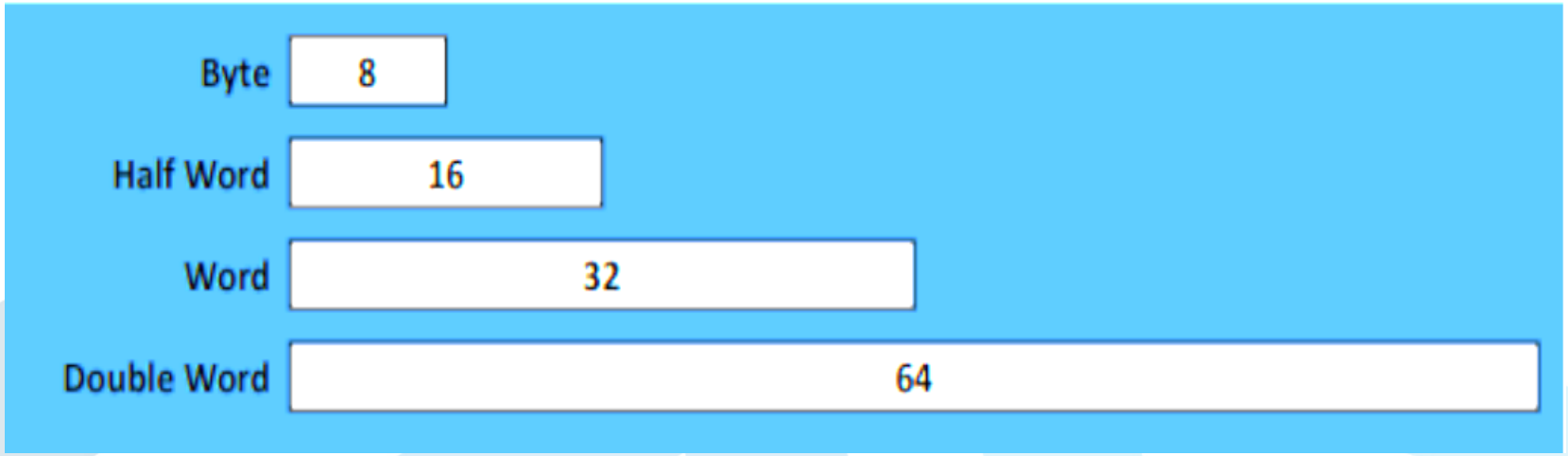
مثال رمزي: ADD R1, R2, R3 يعني R1 = R2 + R3

- صيغة المعاملين: يكون أحد المعاملين مصدراً ونتيجة للعملية في الوقت نفسه.

مثال رمزي: ADD R1, R2 يعني R1 = R1 + R2

Number of memory addresses	Maximum number of operands allowed	Type of architecture	Examples
0	3	Load-store	Alpha, ARM, MIPS, PowerPC, SPARC, SuperH, TM32
1	2	Register-memory	IBM 360/370, Intel 80x86, Motorola 68000, TI TMS320C54x
2	2	Memory-memory	VAX (also has three-operand formats)
3	3	Memory-memory	VAX (also has two-operand formats)

أحجام التخزين للأعداد الصحيحة



التحويل بين أنظمة العد

التحويل بين أنظمة العد

$(16)_{10} = (10)_{16} = 10_{hex}$

$(32)_{10} = (20)_{16} = 20_{hex}$

أمثلة

$(42)_{10} = (32)_{10} + (10)_{10}$

$= (20)_{16} + (A)_{16} = (2A)_{16} = 2A_{hex}$

$2A_{hex} = (0010\ 1010)_2$

$(2A)_{hex} = (20)_{hex} + (A)_{hex}$

$= (32)_{10} + (10)_{10} = (42)_{10}$

4-bit values Binary	Hexadecimal	Unsigned Decimal	Signed Decimal
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	8	8	-8
1001	9	9	-7
1010	A	10	-6
1011	B	11	-5
1100	C	12	-4
1101	D	13	-3
1110	E	14	-2
1111	F	15	-1

التحويل بين أنظمة العد

مثال: حوّل العدد $(100101)_2$ من النظام الثنائي إلى النظام العشري.

الحل:

1 0 0 1 0 1

↓ ↓ ↓ ↓ ↓ ↓

32 16 8 4 2 1

---

32 0 0 4 0 1

- إيجاد حاصل جمع الأرقام المكونة للناتج النهائي من الخطوة السابقة:

$37 = 32 + 0 + 0 + 4 + 0 + 1$

- كتابة الناتج بالنظام العشري:

$(37)_{10} = (100101)_2$

4-bit values Binary	Hexadecimal	Unsigned Decimal	Signed Decimal
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	8	8	-8
1001	9	9	-7
1010	A	10	-6
1011	B	11	-5
1100	C	12	-4
1101	D	13	-3
1110	E	14	-2
1111	F	15	-1

التحويل بين أنظمة العد

حوّل العدد $(AB5)_{16}$ من النظام السداسي عشر إلى النظام الثنائي.

الحل:

- استبدال كل رقم من الأرقام المكونة للعدد بمكافئها المكون من 4 خانات في النظام الثنائي:

A B 5

↓ ↓ ↓

1010 1011 0101

- كتابة الناتج بالنظام الثنائي:

$$_2(1010\ 1011\ 0101) = (AB5)_{16}$$

4-bit values Binary	Hexadecimal	Unsigned Decimal	Signed Decimal
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	8	8	-8
1001	9	9	-7
1010	A	10	-6
1011	B	11	-5
1100	C	12	-4
1101	D	13	-3
1110	E	14	-2
1111	F	15	-1

التحويل بين أنظمة العد

مثال: حوّل العدد $(1011110011)_2$ من النظام الثنائي إلى النظام السداسي عشر.

الحل:

- تقسيم الرقم الثنائي إلى مجموعات يتكون كل منها من 4 خانات من اليمين إلى اليسار، وفي حال كانت المجموعة الأخيرة غير مكتملة يُضاف في نهايتها أصفار حتى تصبح مكونة من 4 خانات:

0010 1111 0011

- استبدال كل مجموعة بمكافئها في النظام السداسي عشر، وفق الجدول الوارد في فقرة التحويل من النظام السداسي عشر إلى النظام الثنائي:

0010 1111 0011

↓ ↓ ↓

2 ،F 3

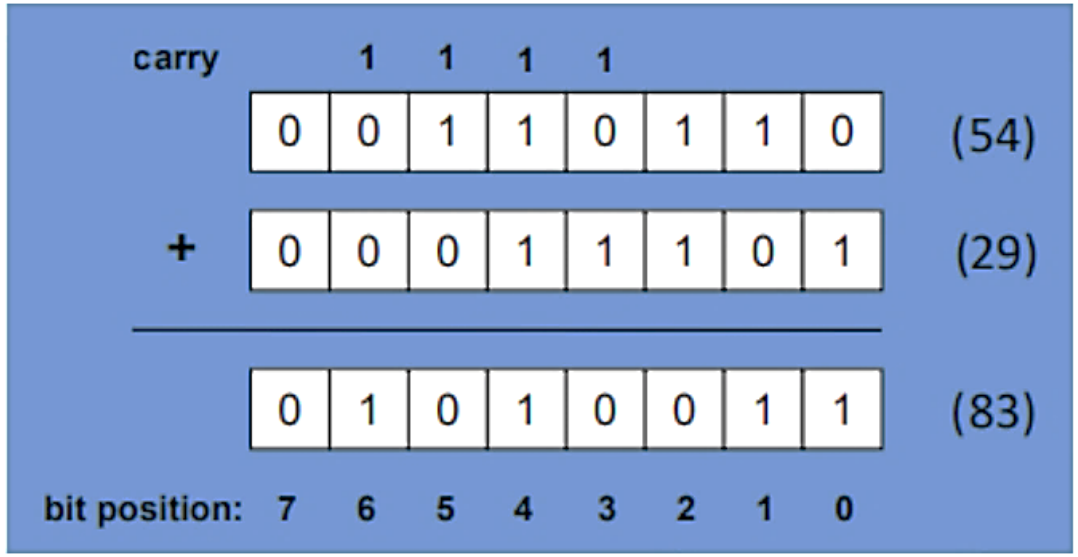
- كتابة الناتج بالنظام السداسي عشر كالآتي:

$$(2F3)_{16} = (1011110011)_2$$

4-bit values

Binary	Hexadecimal	Unsigned Decimal	Signed Decimal
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	8	8	-8
1001	9	9	-7
1010	A	10	-6
1011	B	11	-5
1100	C	12	-4
1101	D	13	-3
1110	E	14	-2
1111	F	15	-1

الجمع في النظام الثنائي



- نبدأ بالبت الأقل أهمية (بت أقصى اليمين)
- نجمع كل زوج من البتات
- نقوم بتضمين الحمل في عملية الجمع، إن وجد

الجمع في النظام السداسي عشري

- نبدأ بالأرقام السداسية العشرية الأقل دلالة
- بفرض Sum = جمع رقمين سداسي عشري
- إذا كان المجموع أكبر من أو يساوي 16
- Sum = Sum - 16 و ينتج لدينا حمل =1
- مثال:

```
carry:        1  1     1
      1 C 3 7 2 8 6 A
    + 9 3 9 5 E 8 4 B
  -----------------------
      A F C D 1 0 B 5
```

A + B = 10 + 11 = 21
بما أن 21 ≥ 16
Sum = 21 – 16 = 5
Carry = 1

الأعداد الصحيحة بإشارة
Signed Integers

توجد عدة طرق لتمثيل الأرقام بإشارة (الأرقام الموجبة و السالبة) :

- المتمم الأحادي 1's complement
- المتمم الثنائي 2's complement

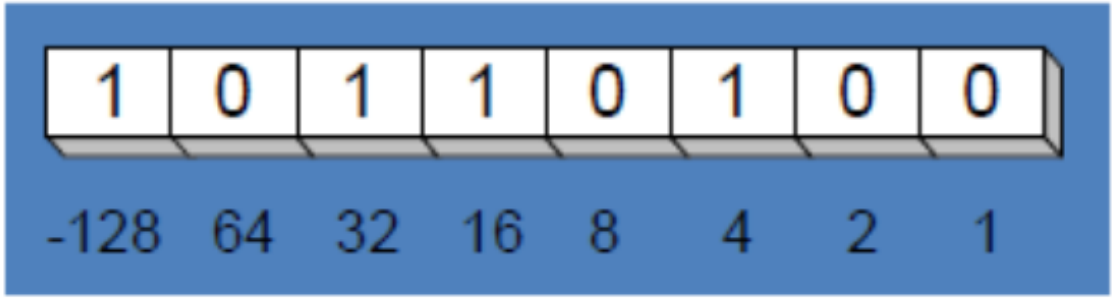
- في الأعداد بإشارة ينقسم المجال إلى جزأين متساويين:
  - الجزء الأول يمثل الأعداد الموجبة
  - الجزء الثاني يمثل الأعداد السالبة
- سيكون التركيز على تمثيل الأعداد بإشارة من خلال المتمم الثنائي باعتبار أنه هو المستخدم على نطاق واسع في المعالجات لتمثيل الأعداد الصحيحة بإشارة .

التمثيل باستخدام المتمم الثنائي

8-bit Binary value	Unsigned value	Signed value
00000000	0	0
00000001	1	+1
00000010	2	+2
. . .	. . .	. . .
01111110	126	+126
01111111	127	+127
10000000	128	-128
10000001	129	-127
. . .	. . .	. . .
11111110	254	-2
11111111	255	-1

- لإيجاد مطال القيمة بإشارة ممثلة باستخدام المتمم

الثنائي نقوم بإسناد وزن سلبي للبت الأكثر أهمية MSB



-128 + 32 + 16 + 4 = -76

تشكيل المتمم الثنائي

- مثال: مثل العدد (-36)بحسب المتمم الثنائي؟

00100100 = +36	للحصول على العدد -36 ننطلق في البداية من العدد +36
11011011	الخطوة1: نعكس الخانات (إيجاد المتمم الأحادي)
+ 1	الخطوة 2: نجمع 1 إلى القيمة الناتجة من الخطوة1
11011100 = -36	حاصل الجمع = التمثيل بحسب المتمم الثنائي

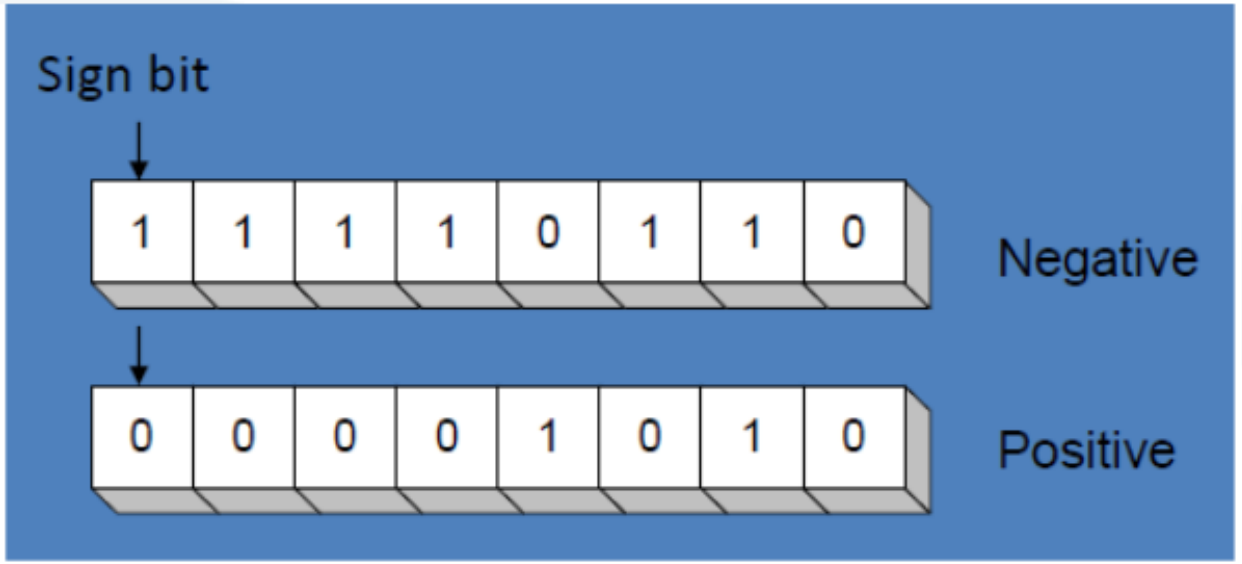
- توجد طريقة أخرى للحصول على المتمم الثنائي:
  - ✓نبدأ بالبحث من الخانة الأقل أهمية عن أول ورود لـ 1
  - ✓نترك كل الأصفار على يمينه بدون تغيير
  - ✓نقلب كل الخانات على يساره (نجعل الأصفار واحدات و بالعكس)

```
Binary Value
= 00100100
2's Complement
= 11011100
```

خانة الإشارة
Sign Bit

1 = negative

0 = positive



توسعة خانة الإشارة

- يمكن توسعة العدد بإشارة من خلال تكرار خانة الإشارة في أقصى اليسار بعدد خانات التوسعة المطلوبة دون أن يؤثر ذلك على إشارة و مطال العددج
- مثال: وسع العدد 10110011 بحيث يصبح طوله 16 بت؟

10110011 = -77 ⟹ 11111111 10110011 = -77

- مثال : وسع العدد 01100010 بحيث يصبح طوله 16 بت؟

01100010 = +98 ⟹ 00000000 01100010 = +98

الطرح الثنائي

- يمكن باستخدام التمثيل بالمتمم الثنائي تحويل عملية الطرح إلى عملية جمع كما يلي :

$$A - B = A + (-B)$$

يتم تجاهل الحمل الأخير

borrow: 1 1 1 carry: 1 1 1 1

$$\begin{array}{r} 01001101 \\ -\ 00111010 \\ \hline 00010011 \end{array} \quad \Rightarrow \quad \begin{array}{r} 01001101 \\ +\ 11000110 \\ \hline 00010011 \end{array}$$

المتمم الثنائي

حصلنا على نفس النتيجة

الحمل و الطفحان
Carry and Overflow

- الحمل مهم عندما :
  - ✓جمع أو طرح الأعداد الصحيحة بدون إشارة
  - ✓يشير إلى أن المجموع خارج المجال
  - ✓إما المجموع أصغر من 0 أوالمجموع أكبر من الحد الأقصى لقيمة n بت بدون إشارة
- الطفحان مهم عندما :
  - ✓جمع أو طرح الأعداد الصحيحة بإشارة
  - ✓يشير إلى أن المبلغ الموقع خارج المجال
- يحدث الطفحان عند:
  - ✓جمع رقمين موجبين ويكون المجموع سالب
  - ✓جمع رقمين سالبين ويكون المجموع موجبًا
- يمكن التخلص من حالة الطفحان بزيادة عدد الخانات الممثلة للأعداد بالمتمم الثنائي

Examples

			1						
	0	0	0	0	1	1	1	1	15
+	0	0	0	0	1	0	0	0	8
	0	0	0	1	0	1	1	1	23

Carry = 0 Overflow = 0

	1	1	1	1	1				
	0	0	0	0	1	1	1	1	15
+	1	1	1	1	1	0	0	0	248 (-8)
	0	0	0	0	0	1	1	1	7

Carry = 1 Overflow = 0

		1							
	0	1	0	0	1	1	1	1	79
+	0	1	0	0	0	0	0	0	64
	1	0	0	0	1	1	1	1	143 (-113)

Carry = 0 Overflow = 1

	1				1	1			
	1	1	0	1	1	0	1	0	218 (-38)
+	1	0	0	1	1	1	0	1	157 (-99)
	0	1	1	1	0	1	1	1	119

Carry = 1 Overflow = 1

جَـامعة
المَـنارة
MANARA UNIVERSITY

نهاية المحاضرة الأولى