1.2 أنظمة العدّ: *Number Systems*

بيّنا في الفصل الأول أنّ أنظمة الحاسب تُعالِج المعلومات داخلياً بعد تحويلها إلى الشكل الرقمي. نظام العدّ الشائع والمستخدم في حياتنا اليومية هو النظام العُشري (المبني على الرقم 10)، يستخدم هذا النظام الأرقام من 0 حتى 9 لتمثيل الأعداد العشرة الأولى الصحيحة ويستخدم بعدها قيم الموضع ليزيد هذا النظام من قيمة الأعداد. فمثلاً العدد 3567.89 يمكن كتابته كالتالي:

$$\mathbf{3567.89} = \underset{1000s}{3\times10^{3}} + \underset{100s}{5\times10^{2}} + \underset{10s}{6\times10^{1}} + \underset{1s}{7\times10^{0}} + \underset{0.1s}{8\times10^{-1}} + \underset{0.01s}{9\times10^{-2}}$$

$$= 3000 + 500 + 60 + 7 + 0.8 + 0.09$$

بالحركة نحو اليسار فإن قيمة كل موضع تساوي عشر مرات قيمة الموضع السابق. تستخدم أنظمة الحاسب الحديثة نظام عد مختلف يدعى الثنائي *binary* وهو مبني على الرقم 2 لتمثيل الأعداد داخليا. سبب استخدامها لهذا النظام هو أنه يبسط تصميم وبناء الدارات الإلكترونية حيث يستخدم القيمتين 0 و1. الدارة الإلكترونية التي تمثل الخانة الثنائية بحاجة إلى حالتين صحيحتين فقط بينما في النظام العشري فيجب امتلاكها لعشرة حالات، في النظام الثنائي فإن قيم الموضع تزداد بمعامل قدره 2 كلما تحركت نحو اليسار والمثال التالي يوضح ذلك:

$$(10100101)_2 = \underset{128s}{1\times2^{7}} + \underset{64s}{0\times2^{6}} + \underset{32s}{1\times2^{5}} + \underset{16s}{0\times2^{4}} + \underset{8s}{0\times2^{3}} + \underset{4s}{1\times2^{2}} + \underset{2s}{0\times2^{1}} + \underset{1s}{1\times2^{0}}$$

$$= 128 + 32 + 4 + 1 = (145)_{10}$$

في الحساب فإنّ مصطلح رقم ثنائي *binary digit* يختصر إلى *bit* ويطلق عليه خانة. كما يمكننا أيضاً تمثيل الكسور بالثنائي وذلك بوضع خانات بعد رمز الفاصلة (ليس فاصلة عشرية *decimal point* لكن فاصلة ثنائية *bicimal point*)، كما هو مبيّن بالشكل التالي:

$$(101.011)_2 = \underset{4s}{1\times2^{2}} + \underset{2s}{0\times2^{1}} + \underset{1s}{1\times2^{0}} + \underset{0.5s}{0\times2^{-1}} + \underset{0.25s}{1\times2^{-2}} + \underset{0.125s}{1\times2^{-3}}$$

$$= 4 + 1 + 0.25 + 0.125 = (5.375)_{10}$$

لاحظ أنّ هذا التمثيل يدعى ذو الفاصلة الثابتة *fixed point* لأنّ الحاسب سيخصص عدداً ثابتاً من الخانات للجزء الصحيح وآخر للجزء الكسري والفاصلة الثنائية ثابتة بين هذين الجزأين.

هناك نظامان آخران شائعان ومستخدمان في العمليات الحسابية هما الثماني (ذو الأساس 8) *octal* والست عشري (ذو الأساس 16) *hexadecimal*. ميزة هذان النظامان هي أنّهما يسمحان بتمثيل الأعداد الثنائية وبشكل مختصر، ففي النظام الثماني نحتاج لثلاثة خانات ثنائية *3 bits* كي تمثل خانة ثمانية واحدة (واحد من الأعداد 0- 7) ونحن بحاجة إلى أربع خانات ثنائية كي تمثل خانة ستة عشرية واحدة (واحد من الأعداد 0- 15).

نستخدم في النظام الست عشري 15 عدداً مختلفاً. وهذه مُعضلة حيث يجب أن نعبر عن الأعداد التي تزيد عن 9 في النظام الست عشري. لتجنب هذه المعضلة وكتابة الأعداد الست عشرية مهما بلغ حجمها فإننا نوظّف الأحرف من A حتى F لتمثيل الأعداد من 10 حتى 15 على التوالي.

يُبيّن الجدول التالي آلية تمثيل الأعداد من 0 حتى 15 بالثنائي والثماني والست عشري:

العشري	الثنائي	الثماني	الست عشري
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	10
11	1011	13	11
12	1100	14	12
13	1101	15	13
14	1110	16	14
15	1111	17	15

يمثل الرقم العشري 197 بالثنائي بالشكل $(11000101)_2$ ويُتحقق منه كالتالي:

$$(11000101)_2 = 1\times2^7 +1\times2^6 + 0\times2^5 +0\times2^4 + 0\times2^3 +1\times2^2 + 0\times2^1 + 1\times2^0=(197)_{10}$$

ويمثل بالثماني بالشكل: $(305)_8$ ويتأكد من ذلك كالتالي:

$$(305)_8 = 3\times8^2 + 0\times8^1 + 5\times8^0=(197)_{10}$$

وبالمثل المكافئ الست عشري هو: $(C5)_{16}$ ويحسب كالتالي:

$$(C5)_{16} = C\times16^1 + 5\times16^0 = 12\times16 +5=(197)_{10}$$

لاحظ أنه غالباً ما نضع العدد ضمن قوسين ونتبعه بلاحقة سفلية كي تشير إلى قاعدة (أساس، جذر) نظام العد وبدون هذا الأمر فإننا لا نستطيع التمييز بين الـ 100 العشرية وبين الـ 100 الثنائية والتي تساوي أربعة.

2.2 التحويل بين أنظمة العد: *Conversion between Number Systems*

يشرح هذا المقطع طرق التحويل بين أنظمة العد المختلفة باليد مع أن الآلات الحاسبة الحالية قادرة على إجراء هذه العملية مباشرة.

1.2.2 التحويل من الثنائي إلى الثماني: *From Binary to Octal*

يمكن تمثيل كل خانة ثمانية بثلاث خانات ثنائية ونبدأ العمل من اليمين إلى اليسار (مبتدئين من الفاصلة الثنائية). وبالنسبة للأعداد التي تحتوي على خانات بعد الفاصلة الثنائية فإننا نعالج الجزء الكسري من اليسار إلى اليمين.

مثال: حوّل العدد الثنائي التالي 10100010101 إلى مكافئه الثماني.

نقوم بتجميع الخانات في مجموعات ثلاثية الخانة منطلقين من اليمين نحو اليسار ويمكننا إلحاق صفراً في الخانة الأكثر أهمية إذا اقتضى الأمر.

خانة ملحقة

Binary	*010*	*100*	*010*	*101*
Octal digits	*2*	*4*	*2*	*5*

مثال: حوّل العدد الثنائي التالي 1101111.0101111 إلى نظيره الثماني.

خانات ملحقة خانات ملحقة

Binary 00 1 101 111. 010 111 1 00
Octal digits 1 5 7 . 2 7 4

$$Result = (157.274)_8$$

2.2.2 التحويل من الثنائي إلى الست عشري:

From Binary to Hexadecimal

يتم عادة تمثيل كل خانة ست عشرية بأربع خانات ثنائية وطريقة التحويل مشابهة لتلك المستخدمة في النظام الثماني.

مثال: حوّل العدد الثنائي $(10100010101)_2$ إلى مكافئه الست عشري.

خانة ملحقة

نقوم بالتجميع إلى مجموعات رباعية 0101 0001 0101

Hexadecimal 5 1 5 $= (515)_{16}$

مثال: حوّل العدد الثنائي ذو الجزء الكسري $(1101111.01011110)_2$ إلى مكافئه الست عشري.

نقوم بالتجميع ومن ثم نستبدل كل أربع خانات ثنائية بخانة ست عشرية مقابلة

0110 1111. 0101 1110

Hexadecimal 6 F . 5 E $= (6F.5E)_{16}$

3.2.2 التحويل من العشري إلى الثنائي: *From Decimal to Binary*

للتحويل من العشري إلى الثنائي نقوم بالقسمة المتكررة للعدد المعطى على الرقم 2 (أساس نظام العد المراد التحويل إليه) حتى نحصل على الصفر النهائي. بعد كل عملية قسمة يعطينا الباقي الخانة التالية في العدد الثنائي.

مثال: حوّل العدد العشري 178 إلى العدد الثنائي المقابل.

المقسوم	باقي القسمة على 2	النتيجة
178	*Is even so 0* زوجي	0
89	*Is odd so 1* فردي	10
44	*Is even so 0* زوجي	010
22	*Is even so 0* زوجي	0010
11	*Is odd so 1* فردي	10010
5	*Is odd so 1* فردي	110010
2	*Is even so 0* زوجي	0110010
1	*Is odd so 1* فردي	10110010
0	*Finish* النهاية	

لذلك فإنّ: $(178)_{10} = (10110010)_2$

لتحويل الجزء الكسري من العشري إلى الثنائي فإنّنا نقوم بعملية الضرب بـ 2 فإذا كانت النتيجة فردية فإنّنا نضيف واحد إلى يمين الجزء الكسري الثنائي، أمّا إذا كانت النتيجة زوجية فإنّنا نضيف صفراً. نوقف عملية التحويل عند عدم وجود جزء كسري متبقي.

مثال: حول العدد العشري $(42.6875)_{10}$ إلى الثنائي المقابل.

الحل: آلية تحويل الجزء الصحيح تتم كما في المثال السابق وهي تساوي:

$$(42)_{10} = (101010)_2$$

وآلية تحويل الجزء الكسري تتم كالتالي: نبدأ بالجزء الصحيح ونضرب بـ 2

المضروب		العدد الثنائي حتى الآن
42.6875		*101010.*
85.375	*Is odd so add a 1*	101010.1
170.75	*Is even so add a 0*	101010.10
341.5	*Is odd so add a 1*	101010.101
683	*Is odd so add a 1*	101010.1011

ولذلك فإن $(24.6875)_{10} = (101010.1011)_2$

ملاحظة: هناك بعض الأعداد التي يمكن أن تمثل بسهولة بالعشري وتعطي كسر تكراري عند تحويلها إلى الثنائي (مثلاً حول العدد 0.2 إلى الثنائي).

ملاحظة: كي تحول من الثنائي إلى العشري فقط اجمع قوى الأساس 2 المقابلة لموضع الخانة كما شرح.

أمثلة إضافية:

حول العدد $(100)_{10}$ إلى الثنائي ومن ثم إلى الست عشري وحول الناتج الست عشري مرة أخرى إلى العشري كي تتأكد من النتيجة.

الحل:

100	*Is even so 0*	0
50	*Is even so 0*	00
25	*Is odd so 1*	100
12	*Is even so 0*	0100
6	*Is even so 0*	00100
3	*Is odd so 1*	100100
1	*Is odd so 1*	1100100
0	*Finish*	

ولهذا يكون:

$$(100)_{10} = (1100100)_2$$

والآن لنحول الناتج إلى الست عشري:

$$(1100100)_2 = \underset{6}{0110}\ \underset{4}{0100} = (64)_{16}$$

لذا فإنَّ:

$$(100)_{10} = (64)_{16}$$

وللتحويل المعاكس الآن من الست عشري إلى العشري:

$$(64)_{16} = 6\times16^1 + 4\times16^0 = 6\times16+4 = 96+4 = 100$$

مثـــال: حوّل العدد $(3.1415)_{10}$ إلى الثنائي وبامتداد قدره ثمانية مواضع واحسب الخطأ المصحوب بهذه العملية.

المضروب		العدد الثنائي
3.1415		*11.*
6.283	*Is even so add a 0*	11.0
12.566	*Is even so add a 0*	11.00
25.132	*Is odd so add a 1*	11.001
50.264	*Is even so add a 0*	11.0010
100.528	*Is even so add a 0*	11.00100
201.056	*Is odd so add a 1*	11.001001
402.112	*Is even so add a 0*	11.0010010
804.224	*Is even so add a 0*	11.00100100

لذا فإن:

$$(3.1415)_{10} = (11.00100100)_2$$

$$(11.001001)_2 = 1\times2^1 + 1\times2^0 + 1\times2^{-3} + 1\times2^{-6}$$
$$= 2 + 1 + 1/8 + 1/64 = 3.140625$$
$$Error = 3.1415 - 3.140625 = 0.000875 \text{ or } 0.02\ \%$$

3.2 تمثيل المعلومات الرقمية:

The Representation of Numerical Information

1.3.2 الأعداد الصحيحة: *Integers*

كي تخزن رقماً داخل الحاسب يجب أن يُحوّل أولاً إلى الثنائي ومن ثم يكتب إلى موقع الذاكرة. ما درسناه سابقاً هو تمثيل الأعداد الكليّة (0، 1، 2، ... الخ) وأجزاؤها. مثلاً كي تمثل الرقم العشري 19 في موقع ذاكرة ذو ثمانية خانات فإننا يجب أن نخزن العدد *00010011*. تتطلب العديد من التطبيقات تمثيل القيم السالبة والموجبة، مثلاً لو أنّنا نودّ تخزين رصيد شخص ما في البنك وكان هذا الشخص مدين للبنك، فإنّنا يجب أن نُمثّل مثل هذا الرصيد بالقيم ذات الإشارة وسوف ندرس تقنيتين مهمّتين وشائعتين

مستخدمتين لتمثيل الأعداد ذات الإشارة هما تقنية الإشارة والطويلة *sign and magnitude* وتقنية المتمم الثنائي *two's complement*.

تقنية الطويلة والإشارة: *Sign and Magnitude*

في هذه التقنية تُعبّر الخانة الأكثر أهمية للعدد عن الإشارة، وتُدعى خانة الإشارة وتأخذ قيمة صفر في العدد الموجب وقيمة واحد في العدد السالب، وتستخدم باقي الخانات لتخزين قيمة (أو مطال) العدد. وهذه أمثلة بسيطة باستخدام التمثيل ذو الـ *16* خانه.

الثنائي	العشري
1000000000000100	*-4*
0000000001100100	*+100*
1000000000010100	*-20*
1000000001100100	*-100*

لاحظ أن للصفر في هذه الطريقة تمثيلان (*0+* و0-) وهذا يسبب أخطاءً في نظام لا يأخذ هذا بعين الاعتبار لأن *0-* هو ليس *0+* ثنائياً.

المتمم الثنائي: *Tow's Complement*

في المتمم الثنائي كما هو الحال في الإشارة والطويلة تمثل الخانة الأكثر أهمية الإشارة. بالنسبة للأعداد الموجبة بالمتمم الثنائي فإن التمثيل مطابق للمطال والإشارة أما الأعداد السالبة فتخزن كمتمم ثنائي للقيمة الموجبة المكافئة. لإيجاد المتمم الثنائي لعدد، قم بعكس (تمّم) كل الخانات (مستبدلاً الواحدات بالأصفار وبالعكس) ومن ثم أضف واحداً إلى الخانة الأقل أهميّة.

في المثال التالي نحسب قيمة *20-* بالمتمم الثنائي وبثمان خانات

+20 = 00010100
Complement = 11101011
+1

-20 = 11101100

كمثال آخر: مثّل *100-* بالمتمم الثنائي ذو الثمان خانات.

+100 = 01100100
Complement = 10011011
+1

-100 = 10011100

للمتمم الثنائي ميزتين مقارنة مع تمثيل الإشارة والمطال، الأولى هي أن هناك تمثيلاً واحداً للصفر والثانية هي أن الأعداد السالبة يمكن أن تضاف مباشرة إلى الأعداد الموجبة معطية نتيجة صحيحة (وهذا غير ممكن في التمثيل ذو الإشارة والطويلة).

مثال: اجمع مايلي: *30+20-*

11101100	*-20*
00011110	*+30*
00001010	$=(10)_{10}$

لأنّنا نتعامل مع أعداد بثمانية خانات؛ فإنّنا نتجاهل المنقول في الخانة التاسعة. مجال الأعداد الممكن تخزينه يعتمد على عدد الخانات المتاحة وهل العدد ذو إشارة أو بدون إشارة.

بالنسبة للأعداد الصحيحة بدون إشارة *unsigned integers* المجال هو من *0* حتى 2^{N-1} وبالنسبة للأعداد الصحيحة ذات الإشارة فإن المجال هو من -2^{N-1} حتى $+2^{N-1}-1$ ، والجدول التالي يبين مجال التخزين لبعض الحجوم المختلفة من خزان الذاكرة.

مجال الأعداد الصحيحة:

حجم التخزين	القيمة الصغرى	القيمة العظمى
8 خانات بدون إشارة	*0*	*255*
8 خانات بإشارة	*-128*	*+127*
16 خانة بدون إشارة	*0*	*65535*
16 خانة بإشارة	*-32768*	*+32767*
32 خانة بإشارة	-2147483648	2147483647

عند إضافة الأعداد إلى بعضها يجب أن يعلم برنامج الحاسب بإمكانية الطفحان *overflow*، والذي يحدث إذا لم يكن ممكناً تخزين النتيجة في المساحة المتاحة. فمثلاً، عند إضافة أعداد ذات ثمانية خانات بدون اشارة مثل *200* و*100* فالناتج هو *300* وهذا لا يمكن تخزينه في موقع ذو ثمانية خانات كما هو مُبيّن في الجدول أعلاه لأن ناتج الجمع هنا أكبر من *255* والناتج سيكون ذو طفحان. إنَّ وضع 1 في خانة راية المنقول بعد عملية الجمع يشير إلى حدوث طفحان بدون إشارة.

في العمليات الحسابية ذات الإشارة تكون القواعد مختلفة، يحدث الطفحان عند جمع عددين لهما نفس الإشارة، وكانت النتيجة ذات إشارة مختلفة. مثلاً جمع *20000-* إلى *30000-* في مواقع ذات إشارة وبطول *16* خانة يُعطي ما يلي:

-20000	*1011000111100000*
+ *-30000*	*1000101011010000*
=*+15536*	*0011110010110000*

لأننا أضفنا عددين سالبين مع بعض فالنتيجة يجب أن تكون سالبة، لكن النتيجة موجبة ولذا فإنّنا نعلم أنّ طفحان ذو اشارة قد حدث عن طريق الواحد الذي سينتج في خانة المنقول.

الدقَّة الكسرية: *Fractional Accuracy*

يُحدّد عدد الخانات في الجزء الكسري لعدد ما الدّقة الممكن الحصول عليها. الخطأ الأعظمي الذي سنحصل عليه عند تخزين العدد سيكون $\pm 1/(2^{N+1})$ حيث N هو عدد الخانات في الجزء الكسري.

فلو أننا استخدمنا 10 خانات لتخزين العدد فإن الخطأ الأعظمي سيكون $\pm 1/2048$.

البايتات والنيبلات: *Bytes and Nibbles*

تعرف البايت على أنها تجمّع من ثمان خانات وتستخدم كوحدة أساسية لقياس سعة التخزين في الحاسب. ويعرف النيبل *nibble* على أنه مجموعة من أربع خانات ويساوي لخانة ستة عشرية واحدة.

4.2 شيفرات المحرف والرسوم: *Character Codes and Graphics*

لا تُستخدم أنظمة الحاسب فقط للتعامل مع البيانات الرقمية، بل يجب استخدامها للتعامل مع معلومات بأشكال أخرى. سننظر في المقطعين التاليين إلى آلية معالجة المحارف والرسوم في نظام الحاسب.

1.4.2 شيفرات المحرف: *Character Codes*

يُمثّل كل محرف في نظام الحاسب بعدد مختلف أو شيفرة ما وعدد المحارف المختلفة التي يمكن تمثيلها يساوي 2^N حيث N هو عدد خانات الشيفرة المستخدمة. يمكن أن

تستخدم أنظمة شيفرة المحرف لتخزين المعلومات ولإرسالها (على شبكة ما مثلاً). هناك شيفرتان شائعتان لتمثيل المحارف هما:

الشيفرة المعيارية الأمريكية لتبادل المعلومات *ASCII*:

American Standard Code for Information Interchange

استخدم مختلف المصنعين في الأيام الأولى لأنظمة الحاسب أشكال شيفرة مختلفة. مثلاً، استخدمت *IBM* شيفرة *EBCIDIC*. وسبَّب ذلك العديد من المشاكل أثناء محاولة الاتصال بين هذه الأنظمة أو تشاركها بالبيانات والسبب هو أن نظام ما قد يمثل المحرف" *A*" مثلاً بطريقة مختلفة عن تمثيله بنظام آخر. لهذا السبب قام معهد المعايير الوطني الأمريكي *ANSI* بتطوير شيفرة *ASCII* مقدِّما للمصنعين نظام تشفير واحد يمكّنهم أن يعملوا عليه جميعا. يعتبر نظام شيفرة *ASCII* الأكثر انتشاراً واستخداماً اليوم.

تستخدم شيفرة *ASCII* المعيارية *7* خانات لتمثيل كل محرف مما يعطينا *128* شيفرة مختلفة، وهناك أيضا إصدار يدعى *ASCII* الموسّع والذي يستخدم *8* خانات لتمثيل *256* شيفرة. الـ 128 شيفرة الأولى في شيفرة *ASCII* الموسّعة هي نفسها في *ASCII* المعيارية. طوِّرت *ASCII* بالأساس كشيفرة نقل وإرسال، ولهذا السبب فإنّ عدداً من الشيفرات (تدعى شيفرات التحكم) لها معنى خاص ومناسب فقط في نقل المعطيات، وهذا وصف لبعض شيفرات التحكم الأكثر انتشارا:

SOH تشير إلى بداية ترويسة الإرسال (بداية الترويسة) *(start of header)*

STX تشير إلى نهاية الترويسة وبداية صندوق النص *(start of text)*

ETX تشير إلى نهاية صندوق النص *(end of text)*

ACK تشير إلى الإشعار باستقبال رسالة ما بشكل صحيح *(acknowledgement)*

NACK تشير إلى إشعار سلبي *negative ack*؛ أي أن الرسالة قد استقبلت بصورة خطأ.

تُستَخدم شيفرات التحكم هذه في اتفاقيات الاتصال مثل *HDLC*.

يُبيّن الجدول التالي مجموعة محارف *ASCII*:

جدول شيفرة محارف *ASCII*

Code	*Char*	*Code*	*Char*	*Code*	*Char*	*Code*	*Char*
0	*NULL*	*32*	*SP*	*64*	*@*	*96*	*`*
1	*SOH*	*33*	*!*	*65*	*A*	*97*	*A*
2	*STX*	*34*	*"*	*66*	*B*	*98*	*B*
3	*ETX*	*35*	*#*	*67*	*C*	*99*	*C*
4	*EOT*	*36*	*$*	*68*	*D*	*100*	*D*
5	*ENQ*	*37*	*%*	*69*	*E*	*101*	*E*
6	*ACK*	*38*	*&*	*70*	*F*	*102*	*F*
7	*BELL*	*39*	*'*	*71*	*G*	*103*	*G*
8	*BKSP*	*40*	*(*	*72*	*H*	*104*	*H*
9	*HT*	*41*	*)*	*73*	*I*	*105*	*I*
10	*LF*	*42*	*	*74*	*J*	*106*	*J*
11	*VT*	*43*	*+*	*75*	*K*	*107*	*K*
12	*FF*	*44*	*,*	*76*	*L*	*108*	*L*
13	*CR*	*45*	*-*	*77*	*M*	*109*	*M*
14	*SO*	*46*	*.*	*78*	*N*	*110*	*N*
15	*SI*	*47*	*/*	*79*	*O*	*111*	*O*
16	*DLE*	*48*	*0*	*80*	*P*	*112*	*P*
17	*DC1*	*49*	*1*	*81*	*Q*	*113*	*Q*
18	*DC2*	*50*	*2*	*82*	*R*	*114*	*R*
19	*DC3*	*51*	*3*	*83*	*S*	*115*	*S*
20	*DC4*	*52*	*4*	*84*	*T*	*116*	*T*
21	*NAK*	*53*	*5*	*85*	*U*	*117*	*u*
22	*SYNC*	*54*	*6*	*86*	*V*	*118*	*v*
23	*ETB*	*55*	*7*	*87*	*W*	*119*	*w*
24	*S0*	*56*	*8*	*88*	*X*	*120*	*x*
25	*S1*	*57*	*9*	*89*	*Y*	*121*	*y*
26	*S2*	*58*	*:*	*90*	*Z*	*122*	*z*
27	*ESC*	*59*	*;*	*91*	*[*	*123*	*{*
28	*S5*	*60*	*<*	*92*	**	*124*	*|*
29	*S5*	*61*	*=*	*93*	*]*	*125*	*}*
30	*S6*	*62*	*>*	*94*	*^*	*126*	*~*
31	*S7*	*63*	*?*	*95*	*_*	*127*	*DEL*

الشيفرة العالمية: *Universal Code (Unicode)*

يصبح قصور شيفرة *ASCII* واضحا بشكل كبير عندما نحاول الاتصال بلغات ليست إنكليزية ولا تستخدم الأحرف الرومانية (مثل العربية والروسية والصينية والإغريقية ...وغيرها)، أيضاً، قد تستخدم بعض اللغات الأحرف الرومانية مع إضافة بعض الرموز الخاصة فوق الأحرف (مثل الألمانية، والفرنسية)، هذه الحالات لا يمكن تمثيلها في شيفرة *ASCII*. وحتى نتمكن من معالجة هذه المشاكل، فقد تم تطوير نظام تشفير جديد يدعى *Unicode*، في عام *1991*. مجلس الأعضاء والمتحكمين بهذا المعيار هم *Apple* و *Sun* و *Xerox* و *IBM* و *Microsoft* و *Novell*.

تستخدم شيفرة *Unicode* ستة عشر (*16*) خانة بدلاً من سبعة في *ASCII* وثمانية في *ASCII* الموسّعة، مما يسمح بتمثيل *65536* محرفا مختلفاً. الـ *128* الأولى هي نفسها كما في *ASCII* وهذا يعني توافق عكسي للمعيار مع سابقه. يُبيّن الجدول التالي بعض مجموعات المحارف المتاحة في *Unicode*.

جدول الشيفرة العالمية *Unicode*

اللغات	الاسم	القيمة العظمى	القيمة الصغرى
European	*Latin*	*007F*	*0000*
Greek	*Greek*	*03FF*	*0370*
Hebrew	*Hebrew*	*05FF*	*0590*
Arabic	*Arabic*	*06FF*	*0600*
Gujarati	*Gujarati*	*0AFF*	*0A80*
Thai	*Thai*	*0E7F*	*0E00*
Cherokee	*Cherokee*	*13FF*	*13A0*
Kham	*Khmer*	*17FF*	*1780*
Chinese, Japanese, Korean	*Han*	*FAFF*	*F900*

معيار *Unicode* هو شيفرة المحرف للغة *Java* وهو مدعوم بكثرة من أنظمة التشغيل مثل *Linux* و *Windows* وبرامج تطبيقات مثل *MS office* ولمزيد من المعلومات حول هذا النظام يمكن الدخول إلى الموقع *www.unicode.org*.

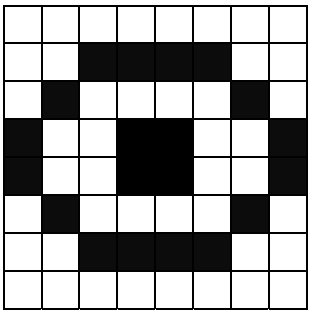
2.4.2 الرسوم: *Graphics*

لتمثيل الرسوم والصور في نظام الحاسب فإنه يتم تجزئتها إلى نقط صغيرة تُدعى الواحدة منها خلية الصورة أو بيكسل *Pixel (picture cell)*. بالنسبة للصورة ذات اللونين الأبيض والأسود، فإن كل خلية صورة (بيكسل) يمكن تمثيلها بخانة واحدة (مثلا 1 يمثل الأبيض و*0* يمثل الأسود). بالنسبة للصور ذات التدرج الرمادي *gray scale*، فإنّه يتم استخدام أكثر من خانة لتمثيل كل خلية صورة. مثلاً باستخدام *8* خانات لكل بيكسل يمكن أن نحصل على *256* مستوى رمادي.

الرسوم الملوّنة: *Color Graphics*

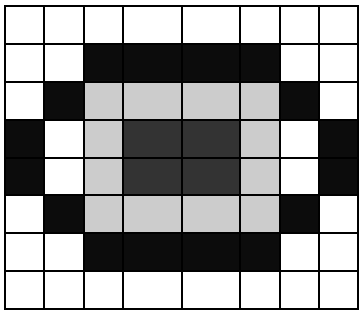
تعتبر الحالة أكثر تعقيداً بالنسبة للرسوم الملونة. الطريقة الأكثر شيوعاً للتمثيل الداخلي هي استخدام مكونات اللون الثلاثة: الأحمر *red* والأخضر *green* والأزرق *blue* (المشار إليها بـ *RGB*) مع أن هناك آليات أخرى. تمزج هذه الألوان الأساسية الثلاثة مع بعضها لتعطي مجال اللون الكلي*full chromatic range*، مثلاً لو أن لدينا أحمر= *255* وأخضر= *0* وأزرق = *255* فإن هذا سوف ينتج اللون البنفسجي *violet*، والشكل التالي يُبيّن أمثلة أكثر.

أ- التمثيل أحادي اللون (الأبيض والأسود)



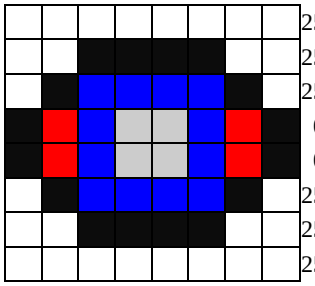
$=11111111_2= 255_{10}$
$=00111100_2=60_{10}$
$=01000010_2=66_{10}$
$=10011001_2=153_{10}$
$=10011001_2=153_{10}$
$=01000010_2=66_{10}$
$=00111100_2=60_{10}$
$=11111111_2=255_{10}$

ب- التمثيل الرمادي:



=255,255,255,255,255,255,255,255
=255,255,0,0,0,0,255,255
=255,0,127,127,127,127,0,255
=0,255,127,0,0,127,255,0
=0,255,127,0,0,127,255,0
=255,0,127,127,127,127,0,
=255,255,0,0,0,0,255,255
=255,255,255,255,255,255,255,255

ج- التمثيل الملون:



r1	g1	b1	r2	g2	b2	r3	g3	b3	r4	g4	b4	r5	g5	b5	r6	g6	b6	r7	g7	b7	r8	g8	b8
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	0	0	0	0	0	0	0	0	0	0	0	0	255	255	255	255	255	255
255	255	255	0	0	0	0	0	255	0	0	255	0	0	255	0	0	255	0	0	0	255	255	255
0	0	0	255	0	0	0	0	255	28	213	227	28	213	227	0	0	255	255	0	0	0	0	0
0	0	0	255	0	0	0	0	255	28	213	227	28	213	227	0	0	255	255	0	0	0	0	0
255	255	255	0	0	0	0	0	255	0	0	255	0	0	255	0	0	255	0	0	0	255	255	255
255	255	255	255	255	255	0	0	0	0	0	0	0	0	0	0	0	0	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255

التمثيل البياني: الأعلى لأحادي اللون والوسط للرمادي والأسفل للملون

يُبيّن الشكل أعلاه كيفية تمثيل صورة بسيطة لعين في الحاسب بالأسود والأبيض وبالرمادي وبالملون.

إذا استخدمنا 8 خانات لتمثيل كل لون فإنّ العدد الإجمالي للألوان الممكنة هو $256\times256\times256=16777216$ (تقريباً 17 مليون لون). هذا الشكل يشار إليه على أنه اللون الحقيقي وهو مدعوم بشكل كبير من قبل مصنعي كروت الفيديو.

5.2 ملخص: *Summary*

تفحّصنا في هذا الفصل كيفية مُعالجة وتمثيل وتخزين أنظمة الحاسب للمعلومات الرقمية وغير الرقمية. حيث استعرضنا بالإضافة إلى مخطط الحاسب المبسّط أنظمة العدّ الشائعة مثل الثنائي والثماني والست عشري وآليات التحويل فيما بينها، ثم انتقلنا للحديث عن آليات التمثيل والشيفرات للمحارف، وأشهر هذه المحارف، وتكلمنا أخيراً عن تمثيل الرسوم والصور.

6.2 تمارين: *Exercises*

1- حول الأعداد العشرية التالية إلى الأعداد الثنائية المقابلة:
131.625, 512.5, 0.4375, 256, 103, 63, 19, 21, 6, 15, 13, 41

2- نفذ عمليات الطرح التالية باستخدام المتمم الثنائي:

a) 00110110-00011101 *c) 01001001-01101000*

b) 00011111-11101010 *d) 00001110-00001111*

3- اجمع بالمتمم الثنائي القيم التالية وحدد هل حدث طفحان بالنسبة لكل منها:
000+001, 000+111, 111+110, 100+111, 100+100

4- اشرح إحدى الطرق المستخدمة بشكل شائع في تمثيل الأعداد ذات الإشارة في الحواسب الرقمية. ما هي القيم العظمى والصغرى الممكن تمثيلها بهذه الطريقة.

5- حول الرسالة التالية إلى أعداد مستخدماً نظام تشفير *ASCII* في النص:
"Hello how are you" لا تنسى الفراغات.

6- هذا مثال عن بعض شيفرات *ASCII* المخزنة في الذاكرة (القيم بالعشري)
65 83 67 73 73 32 109 101 115 115 97 103 101 32 104 101 114 101.
ما هو النص المقابل؟

7- اذكر طريقة مبسطة لتحويل رسالة *ASCII* بأحرف صغيرة إلى رسالة بأحرف كبيرة وبالعكس.