

4 الفصل الرابع

برمجة المعالج intel8085

**Programming The intel8085
Microprocessor**

1-4 مقدمة

لبرمجة أى معالج لا بد من دراسة مجموعة الأوامر الخاصة به ولكى نسهل دراسة هذه الأوامر سنقوم بتقسيمها إلى مجموعات من حيث الوظيفة التى يؤديها كل أمر وسندرس بالتفصيل فى كل مجموعة بعض الأوامر الكثيرة الاستخدام مع التمثيل ببعض الأمثلة ، على أننا سنعرض فى نهاية الفصل لجدول تحتوى على جميع أوامر الشريحة موضوعة فى مجموعات ثم سنعرض أيضا جدولا يحتوى هذه الأوامر مرتبة أبجديا مع نبذة بسيطة عن وظيفة كل أمر وشفرته .

2-4 مجموعة أوامر الانتقال Transfer instructions

يقوم أى أمر من أوامر هذه المجموعة بنقل معلومة من مكان لآخر حيث المكان الذى تخرج منه المعلومة يسمى بالمصدر Source وسنرمز له بالرمز sss وهذا المكان قد يكون مسجلا داخل شريحة المعالج وقد يكون مكانا أو بايت من أماكن الذاكرة . وأما المكان الذى ستذهب إليه المعلومة فسوف نسميه الهدف Destination وسنرمز له بالرمز ddd وهذا المكان أيضا قد يكون مسجلا داخل شريحة المعالج وقد يكون بايت من بايتات الذاكرة كما سنرى . شكل (1-4) يبين أهم الأوامر الموجودة فى مجموعة أوامر الانتقال الخاصة بالشريحة 8085 والتي تهتمنا فى هذه المرحلة من دراسة لغة التجميع .

MOV
MVI
LXI
LDA
STA
LHLD
SHLD

شكل (1-4) بعض أوامر الانتقال الكثيرة الاستخدام للشريحة 8085

1-2-4 الأمر MOV

الصورة العامة لهذا الأمر هي :

MOV ddd,sss

مسجل sss ← مسجل ddd

ومعنى هذا الأمر انقل أو حرك (وهذا هو ترجمة كلمة move) المعلومة الموجودة في المصدر sss إلى الهدف ddd ، لاحظ أن المعالج عندما يقوم بنقل معلومة فإنه ينقل صورة منها فقط أما أصل المعلومة فيظل في المصدر ولا يتغير . الصورة العامة لشفرة هذا الأمر الثنائية يمكن كتابتها كما يلي :

01dddsss

حيث sss تستبدل بشفرة مصدر المعلومة سواء كانت مسجلا أم ذاكرة و ddd تستبدل أيضا بشفرة الهدف الذى ستلجأ إليه المعلومة سواء كان مسجلا أم ذاكرة. لاحظ أن هذا الأمر يتكون دائما من بايت واحدة . راجع شفرات المسجلات في الفصل الثانى ، جدول 1-2 ، وانظر المثال التالى :

مثال 1-4

1. الأمر MOV A,B

شفرته الثنائية هي 01111000

شفرته الست عشرية هي 78H

هذا الأمر سينقل محتويات المسجل B (صورة منها فقط) إلى المسجل A لاحظ أن المسجل B هو المصدر sss ولذلك استبدلناه بشفرته الثنائية وهي 000 والمسجل A هو الهدف ddd واستبدلناه بشفرته الثنائية 111 لتصبح شفرة الأمر الثنائية هي 01111000 أو 78H فى النظام الست عشري ، ولمزيد من الأمثلة إليك ما يلي :

2. الأمر MOV L,D

شفرته الثنائية هي 01101010

شفرته الست عشرية هي 6AH

3. الأمر MOV C,C

شفرته الثنائية هي 01001001

شفرته الست عشرية هي 49H

هذا الأمر ينقل محتويات المسجل C إلى نفسه وهذا يكافئ ، لا تعمل شيئا . مثل هذه الأوامر التى لا تعمل شيئا لها أهمية كبيرة فى الكثير من التطبيقات كما سيأتى فيما بعد .

في جميع الأوامر السابقة كنا ننقل المعلومة من مسجل إلى مسجل آخر ، ماذا لو أردنا نقل معلومة من مسجل إلى الذاكرة أو العكس . المثال التالي سيوضح هذه العملية .

مثال 2-4

1. الأمر MOV M,A

شفرته الثنائية هي 01110111

شفرته الست عشرية هي 77H

هذا الأمر ينقل محتويات المسجل A وهو مصدر المعلومة إلى الذاكرة M وهي الهدف الذي ستذهب إليه المعلومة ، لاحظ أن M استبدلت بالشفرة 110 كما في جدول 1-2 . السؤال الآن هو : في أي مكان أو في أي عنوان في الذاكرة ستذهب محتويات المسجل A ؟ في جميع الأوامر التي تتعامل مع الذاكرة بهذا الشكل يكون العنوان موجودا في زوج المسجلات HL ، أي أن محتويات المسجل A ستذهب إلى بايت الذاكرة التي يوجد عنوانها في المسجلين H و L . لاحظ أن هذه الطريقة هي ما سنسميها بطريقة التعامل غير المباشر مع الذاكرة عندما سنصنف طرق التعامل مع الذاكرة في نهاية هذا الفصل .

2. الأمر MOV B,M

01000110

46H

هذا الأمر سينقل محتويات بايت الذاكرة التي يوجد عنوانها في المسجلين H و L إلى المسجل B .

2-2-4 الأمر MVI

هذا الأمر معناه "انقل المعلومة الفورية" أي Move the Immediate data والصورة العامة له هي :

MVI ddd,data8

ddd ← data8

هذا الأمر يضع المعلومة المكونة من ثمانية بتات (data8) في الهدف ddd . الهدف ddd قد يكون مسجلا أو الذاكرة M كما سنرى في الأمثلة . هذا الأمر يتكون دائما من اثنتين من البايتات . واحدة هي شفرة الأمر operation code واختصارا op code والبايت الأخرى هي المعلومة (data8) . وعلى ذلك ستكون الصورة العامة للشفرة الثنائية لهذا الأمر كالتالي :

00ddd110

data8

حيث ddd تستبدل بشفرة المسجل أو الذاكرة M المراد وضع المعلومة فيها .

مثال 3-4

1. MVI B,53H

00000110 الشفرة الثنائية هي

01010011

06H الشفرة الست عشرية هي

53H

هذا الأمر سيضع المعلومة الفورية أو الثابت 53H في المسجل B . نفضل أن نسمى مثل هذه المعلومة بالمعلومة الفورية لأنها ليس لها مصدرا وإنما مصدرها هو المستخدم نفسه ، ومن هنا كان الحرف I في الأمر وهو اختصارا لكلمة Immediate أو فوري وسوف يصادفنا أوامر أخرى تحتوى الحرف I وكلها تتعامل مع معلومات فورية أو ثوابت بهذا الشكل .

من الممكن تحميل معلومة فورية Immediate في مكان ما في الذاكرة بحيث يكون عنوان هذا المكان في المسجلين HL كالتالى :

2. MVI M,data8

والشفرة الثنائية لهذا الأمر ستكون كالتالى :

00110110

data8

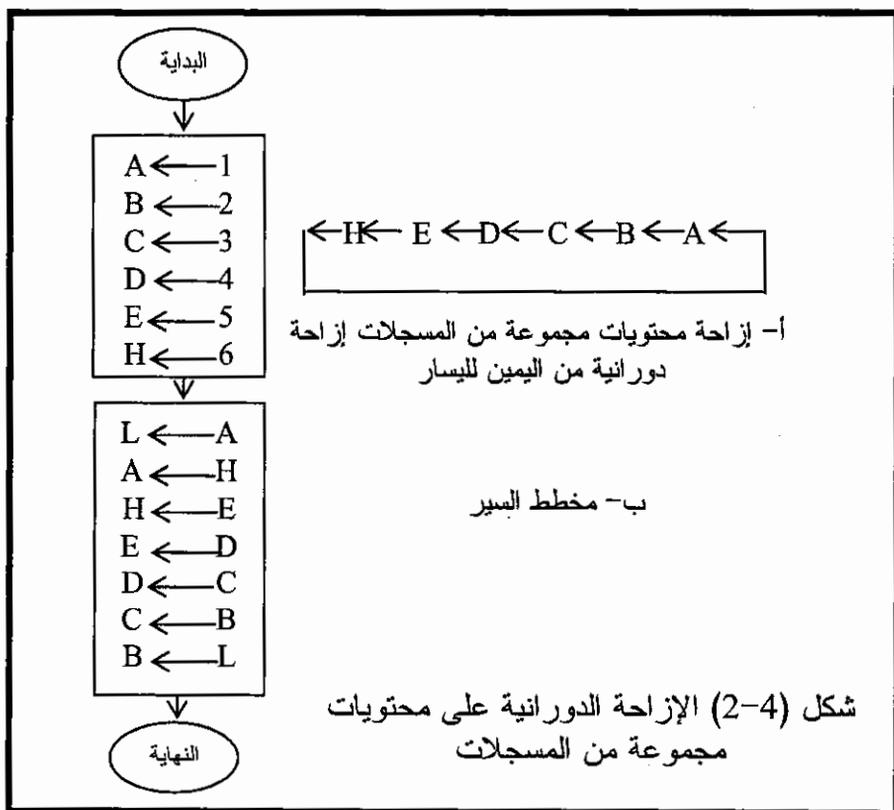
المثال التالى يعتبر تدريبا جيدا على الأمرين MOV و MVI

مثال 4-4

المطلوب تحميل المسجلات A و B و C و D و E و H بالمعلومات الفورية التالية : 01 و 02 و 03 و 04 و 05 و 06 على التوالى ، ثم بعد ذلك يتم عمل إزاحة دورانية لهذه المحتويات كما هو مبين فى شكل (4-2) بحيث أن محتويات المسجل A تذهب إلى المسجل B ومحتويات B تذهب إلى C وهكذا إلى أن تذهب محتويات المسجل H إلى المسجل A دون فقد محتويات أى مسجل من هذه المسجلات . شكل (4-2ب) يوضح مخطط السير لهذا البرنامج ، وشكل (4-3) يبين الشفرات الأسمبلى والثنائية والست عشرية للبرنامج . لتنفيذ هذا البرنامج يمكنك أن تفتح ال RAM عند العنوان E000 وتبدأ فى إدخال البرنامج إما بالشفرات الثنائية أو الشفرات الست عشرية ، وأما إذا كان الميكروكومبيوتر الذى تستخدمه به المجمع (الأسمبلر) الخاص بالشريحة 8085 فيمكنك الدخول فيه وكتابة البرنامج باستخدام شفرات التجميع وسنترك تفاصيل عملية إدخال البرنامج على الجهاز لأنها تختلف على حسب الإمكانيات ومن

شخص لآخر . بعد الانتهاء من إدخال البرنامج يمكنك تنفيذه باستخدام الأمر GO E000 أو من الأفضل تنفيذه بطريقة الخطوة خطوة حيث يمكنك في هذه الحالة متابعة البيانات أثناء انتقالها من مسجل لآخر .

عند استخدام الأمر MOV M,C مثلا لنقل محتويات المسجل C إلى الذاكرة أو الأمر MVI M,33 لتحميل المعلومة الفورية 33 في الذاكرة ذكرنا أن عنوان الذاكرة الذي سيتم التعامل معه يكون دائما في المسجلين HL . السؤال الآن كيف نضع هذا العنوان في المسجلين H و L؟ الإجابة عن هذا السؤال توجد في الأمر LXI .



3-2-4 الأمر LXI

الصورة العامة لهذا الأمر هي:

LXI rp,data16

rp ← زوج المسجلات data16

الشفرة الثنائية للأمر كالتالى :

00rp0001

البايت ذات القيمة الصغرى من المعلومة data16

البايت ذات القيمة العظمى من المعلومة data16

العناوين	شفرات أسمبلى	شفرات ثنائية	شفرات ست عشرية
E000	MVI A,01	00111110	3E
E001		00000001	01
E002	MVI B,02	00000110	06
E003		00000010	02
E004	MVI C,03	00001110	0E
E005		00000011	03
E006	MVI D,04	00010110	16
E007		00000100	04
E008	MVI E,05	00011110	1E
E009		00000101	05
E00A	MVI H,06	00100110	26
E00B		00000110	06
E00C	MOV L,A	01101111	6F
E00D	MOV A,H	01111100	7C
E00E	MOV H,E	01100011	63
E00F	MOV E,D	01011010	5A
E010	MOV D,C	01010001	51
E011	MOV C,B	01001000	48
E012	MOV B,L	01000101	45

شكل (3-4) برنامج الإزاحة الدورية

حيث يقوم هذا الأمر بتحميل زوج المسجلات الذى توضع شفرته بدلا من rp (فى البايت الأولى للأمر) بالمعلومة الفورية data16 أى المكونة من 16 بت والتي توضع فى الإثنين بايت التاليتين . حرف I الموجود فى صورة الأمر له نفس الدلالة التى عرفناها مسبقا والتي تعنى فورى أى Immediate . لاحظ أن المعلومة مكونة من 16 بت وشفرة الأمر op code ستكون بايت واحدة ، لذلك فإن هذا الأمر يتكون دائما من ثلاث بايتات كما سنرى فى الأمثلة . لاحظ أيضا أن هناك أربعة أزواج من المسجلات فقط يمكن استخدامها مع هذا الأمر وهى

الإزواج HL و BC و DE و SP التي سبق أن ذكرنا شفراتها في جدول 2-1. لذلك فإن البايت ذات القيمة العظمى من المعلومة ال16 بت تذهب دائما إلى المسجل ذى القيمة العظمى من الزوج وهو المسجل B أو D أو H أو النصف الأعلى من المسجل SP والبايت ذات القيمة الصغرى تذهب إلى المسجل ذى القيمة الصغرى من الزوج وهو المسجل C أو E أو L أو النصف الأول من المسجل SP .

لتتل 5-4

LXI H,2C3A	1. شفرة الأسمبلى
00100001	الشفرة الثنائية
00111010	
00101100	
21	الشفرة الست عشرية
3A	
2C	

يقوم هذا الأمر بتحميل زوج المسجلات HL بالمعلومة الفورية 2C3A بحيث ستذهب البايت ذات القيمة العظمى وهي 2C إلى المسجل H والبايت ذات القيمة الصغرى وهي 3A إلى المسجل L . بنفس الطريقة يمكن تحميل الأزواج الأخرى من المسجلات . كما رأينا فإن هذا الأمر مهم جدا عند التعامل مع الذاكرة حيث عنوان المكان المراد التعامل معه يوضع في المسجلين H و L باستخدام هذا الأمر .

لتتل 6-4

حمل مكان الذاكرة E100 بالمعلومة 66H . أحد الطرق لكي يتم ذلك هي أن نقوم بتحميل العنوان E100 فى زوج المسجلات HL ثم نستخدم الأمر MVI كما يلي :

E000, E001, E002 LXI H,E100
E003, E004 MVI M,66H

لاحظ أننا استخدمنا هنا الشفرات الأسمبلى فقط ولاحظ أيضا أن الأمر LXI يشغل ثلاث بايتات كما ذكرنا وهي البايتات E000, E001, E002 . إن الطريقة السابقة فى التعامل مع الذاكرة والتي تستخدم زوج المسجلات HL كوسيط يحمل العنوان المراد التعامل معه تعتبر بل وتسمى بالطريقة غير المباشرة فى التعامل مع الذاكرة حيث أن المعالج قبل أن يذهب إلى الذاكرة سواء للقراءة أو للكتابة لابد وأن يمر أولا على زوج المسجلات HL ليعرف

منهما العنوان الذى سيذهب إليه . إن ذلك تماما مثلما أنك تقول لصديقك محمد ياأخى يامحمد وأنت مسافر إلى الرياض خذ هذه الرسالة وأعطها لأخى أحمد ولكنى لا أعرف عنوانه فرجاء أن تذهب إلى أخى الأكبر محمود وهو يسكن معنا هنا فتعرف منه عنوان أحمد قبل أن تسافر إلى الرياض . صديقك محمد فى هذا المثال يمثل المعالج الذى سيقوم بالتنفيذ وأما أخوك الأكبر محمود فيمثل المسجلين HL الذين عندهما عنوان أخيك أحمد الذى يمثل بايت الذاكرة المراد التعامل معها . نعيد التأكيد هنا أن زوج المسجلات HL فقط هما اللذان يحتويان عنوان المكان المراد التعامل معه فى مثل هذه الأوامر .

هناك طريقة أخرى للتعامل مع الذاكرة وهى الطريقة المباشرة حيث يحتوى الأمر نفسه على عنوان مكان الذاكرة المراد التعامل معه ، وإن ذلك مثلما تقول لصديقك محمد ياأخى يا محمد خذ هذه الرسالة وأنت مسافر إلى الرياض وأعطها إلى أخى أحمد ولا تنسى أن عنوان أخى أحمد مكتوب على الرسالة هذه المرة . الأمران اللذان يقومان بهذه المهمة من قائمة أوامر الشريحة 8085 هما الأمران STA و LDA كما يلى :

4-2-4 الأمران STA و LDA

الأمر STA يعنى خزن محتويات المركم Store Accumulator والأمر الثانى يعنى حمل المركم Load Accumulator والصورة العامة للأمرين هى :

STA addr

محتويات المسجل A ← العنوان (addr)

LDA addr

محتويات العنوان (addr) ← المسجل A

والصورة الست عشرية للأمر STA هى :

32

البابيت ذات القيمة الصغرى من العنوان Addr

البابيت ذات القيمة العظمى من العنوان Addr

والصورة الست عشرية للأمر LDA هى :

3A

البابيت ذات القيمة الصغرى من العنوان Addr

البابيت ذات القيمة العظمى من العنوان Addr

الأمر الأول STA سيقوم بتخزين محتويات مسجل التراكم A فى مكان الذاكرة الذى عنوانه فى (الإثنين بايت) التاليتين لشفرة الأمر نفسه وأما الأمر LDA فإنه يقوم بتحميل مسجل التراكم A بمحتويات مكان الذاكرة الذى عنوانه فى (الإثنين بايت) التاليتين لشفرة الأمر نفسه . هناك ملاحظتان هامتان على كل من هذين

الأمرين ، الأولى هي أن هذين الأمرين لا يتعاملان إلا مع مسجل التراكم A فقط ، فانت مثلا إن أردت أن تخزن محتويات المسجل B في أي مكان في الذاكرة بهذه الطريقة المباشرة فلا بد وأن تنتقل محتويات المسجل B إلى المسجل A أولا ثم تستخدم الأمر STA ، وكذلك الحال إن أردت أن تحمل أي مسجل وليكن C مثلا بمحتويات أي مكان في الذاكرة بالطريقة المباشرة فعليك باستخدام الأمر LDA الذي يضع محتويات الذاكرة في مسجل التراكم ثم تقوم أنت بنقل هذه المحتويات من مسجل التراكم إلى أي مسجل آخر وليكن C كما ذكرنا .

الملاحظة الثانية هي أنه طالما أن هذين الأمرين يحتويان على عنوان فلا بد وأن يتكون كل منهما من ثلاث بايتات ، واحدة هي شفرة الأمر ، أي 32 في حالة الأمر STA و 3A في حالة الأمر LDA وأما (الإثنين بايت) الأخرى فتحتوي العنوان المراد التعامل معه كما في الصورة العامة للأمرين . تذكر أن أي عنوان يتكون دائما من 16 بت أي 2 بايت .

مثال 4-7

E000 E001 E002 STA E100

E003 E004 E005 LDA E101

الأمر الأول سيخزن محتويات المسجل A في المكان E100 في الذاكرة والأمر الثاني سيجمل المسجل A بمحتويات المكان E101 من الذاكرة ، لاحظ أنه في ذاكرة البرنامج كل أمر من الأمرين يشغل ثلاث بايتات .

القرار باستخدام أي واحدة من الطريقتين (المباشرة أو غير المباشرة) في عملية البرمجة يتوقف على التطبيق الذي يستخدم فيه هذا البرنامج وسنرجى هذا الموضوع قليلا إلى أن ندرس بعض الأوامر الأخرى وعندها سنوضح أن هناك برامج يفضل فيها استخدام الطريقة المباشرة وأخرى لا بد فيها من استخدام الطريقة غير المباشرة .

4-2-5 الأوامر LHL و SHLD

معناها load H,L Direct أي حمل المسجلين H و L مباشرة ، و Store H,L Direct أي خزن المسجلين H و L مباشرة وكلا الأمرين كما نرى تأثيره عكس الآخر . الصورة العامة لهذين الأمرين هي :

LHLD addr

محتويات (addr) ← H ، محتويات (addr) ← L

SHLD addr

محتويات (addr+1) ← H ، محتويات (addr) ← L

الصورة الست عشرية للأمر LHL D هي :

2A

البايت ذات القيمة الصغرى للعنوان addr

البايت ذات القيمة العظمى للعنوان addr

الصورة الست عشرية للأمر SHLD هي :

22

البايت ذات القيمة الصغرى للعنوان addr

البايت ذات القيمة العظمى للعنوان addr

حيث يقوم الأمر الأول LHL D بتحميل المسجلين H و L بمحتويات العنوان addr والذي يليه في الذاكرة ، أى أن محتويات العنوان addr تذهب إلى المسجل L ومحتويات العنوان الذى يليه addr+1 تذهب إلى المسجل H . الأمر SHLD يقوم بالعملية العكسية للأمر LHL D حيث يقوم بتخزين محتويات المسجلين H و L فى العنوان addr والذي يليه بحيث تذهب محتويات المسجل L إلى العنوان addr ومحتويات المسجل H إلى العنوان addr+1 .

مثال 8-4

1. افترض الوضع التالى فى المسجلين HL ومكانى الذاكرة E100 و E101 :

H	L
89	76

E100	FF
E101	FF

بعد تنفيذ الأمر SHLD E100 سيصبح الوضع كما يلى :

H	L
89	76

E100	76
E101	89

2. افترض الوضع التالى فى المسجلين HL ومكانى الذاكرة E100 و E101 :

H	L
89	76

E100	3A
E101	B5

بعد تنفيذ الأمر LHL D E100 سيصبح الوضع كالتالى :

H	L
B5	3A

E100	3A
E101	B5

إلى هنا سنكتفى بهذه المجموعة من أوامر الانتقال ولمزيد من المعرفة بباقي أوامر الإنتقال انظر الأشكال الموجودة في آخر هذا الفصل والتي تحتوى على نبذة مختصرة عن كل أمر .

3-4 تمارين

1. اكتب برنامجا يضع الأرقام الست عشرية 1 إلى A في عناوين الذاكرة E201 إلى E20A.
2. اكتب برنامجا يضع الرقم 77 في المسجل D والرقم B4 في المسجل C ثم يقوم باستبدال محتويات كل من المسجلين ، أى أن محتويات المسجل D تذهب إلى المسجل C ومحتويات المسجل C تذهب إلى المسجل D دون أن تفقد أيا من محتويات المسجلين ، هذه العملية تسمى Swapping بمعنى استبدال أو مفايضة .
3. نفذ عملية الاستبدال السابقة في المسألة رقم 2 ولكن هذه المرة على محتويات البايث E100 والبايث E101 .
4. اكتب برنامجا للإزاحة الدورانية كالموجود في مثال 4-4 ولكن هذه المرة على محتويات الذاكرة E100 و E101 و E102 و E103 و E104 و E105 .

للإحضاة : يجب عمل مخطط سير لكل برنامج ثم تترجم هذه الخريطة إلى برنامج بلغة التجميع مع الشفرات الثنائية والست عشرية وتحديد العناوين لكل أمر وذاكرة البرنامج وذاكرة البيانات لكل برنامج ، أى أن كل برنامج يحل بنفس طريقة حل المثال رقم 4-4 وحتى تكتمل الفائدة بالذات عند هذا المستوى يجب أن تحمل هذه البرامج بالشفرات الثنائية أولا ثم الشفرات الست عشرية ثم شفرات التجميع .

4-4 مجموعة أوامر الحساب Arithmetic Instructions

ADD
SUB
ADI
SUI
ADC
SBB
INR
INX
DCR
DCX

شكل (4-4) مجموعة أوامر الحساب

سندرس في هذا الجزء بعض الأوامر التي تقوم بإجراء العمليات الحسابية الأولية وهي الجمع والطرح . كما علمنا من قبل فإن مسجل التراكم لابد وأن يكون طرفا في أى عملية من هذه العمليات كما أن نتيجة هذه العملية سواء كانت جمعا أو طرحا تكون دائما موجودة في مسجل التراكم A . هناك أيضا خاصية مهمة في هذه المجموعة من الأوامر غير موجودة في الأوامر الأخرى وهي أن أوامر الحساب (ومثلها أيضا أوامر المنطق) عندما يتم تنفيذ أى أمر منها فإن جميع الأعلام الموجودة في مسجل الحالة SR تتأثر بنتيجة هذه العملية الحسابية أو المنطقية . راجع مسجل الحالة ومحتوياته ومتى يكون أى علم من الأعلام يساوى صفرا أو واحدا وذلك في الفصل الثانى حيث أن جميع الأعلام الموجودة في مسجل الحالة في المعالج 8085 هى نفسها تماما التى شرحت في هذا الجزء (4-4-2) . شكل (4-4) يبين مجموعة الأوامر التى سندرسها هنا حيث فى جميع الأمثلة التى سنستعين بها فى هذا الجزء سنكتفى باستخدام شفرات الأسمبلى فقط على أساس أنه تم التدريب الكافى على الشفرات الثنائية والست عشرية مع مجموعة أوامر الانتقال ومن يريد الاستزادة فى التدريب يمكنه الإستمرار فى ذلك مستعينا بالشفرات الست عشرية التى سنذكرها مع الصورة العامة لكل أمر وكذلك فى الأشكال الموجودة فى نهاية هذا الفصل .

1-4-4 الأمان ADD و SUB

حيث ADD بمعنى إجمع و SUB هي أختصار كلمة SUBtract بمعنى اطرح ،
والصورة العامة لأمر الجمع ADD هي :

ADD reg

$A \leftarrow A + \text{reg}$

حيث سيقوم هذا الأمر بجمع محتويات المسجل reg أو مكان الذاكرة M ، الذي
عنوانه في المسجلين HL ، على محتويات مسجل التراكم A وستوضع نتيجة
الجمع في المسجل A مع التأثير على جميع الأعلام . الشفرة الثنائية العامة لهذا
الأمر هي 10000xxx حيث تستبدل xxx بشفرة المسجل الذي سيجمع مع مسجل
التراكم . كمثال على ذلك الأمر ADD B ستكون شفرته الثنائية هي 10000000
وشفرته الست عشرية هي 80H حيث استبدلت xxx بشفرة المسجل B وهي 000 .

الصورة العامة لأمر الطرح SUB هي :

SUB reg

$A \leftarrow A - \text{reg}$

يقوم هذا الأمر بطرح محتويات المسجل reg أو مكان الذاكرة M ، الذي عنوانه
في المسجلين HL ، من محتويات مسجل التراكم وسوف توضع نتيجة الطرح
في مسجل التراكم مع التأثير على جميع الأعلام . الشفرة الثنائية العامة لهذا
الأمر هي 10010xxx حيث تستبدل xxx بشفرة المسجل المطلوب طرح
محتوياته من مسجل التراكم . كمثال على ذلك الأمر SUB M ستكون شفرته
الثنائية هي 10010110 وشفرته الست عشرية هي 96H .

مثال 9-4

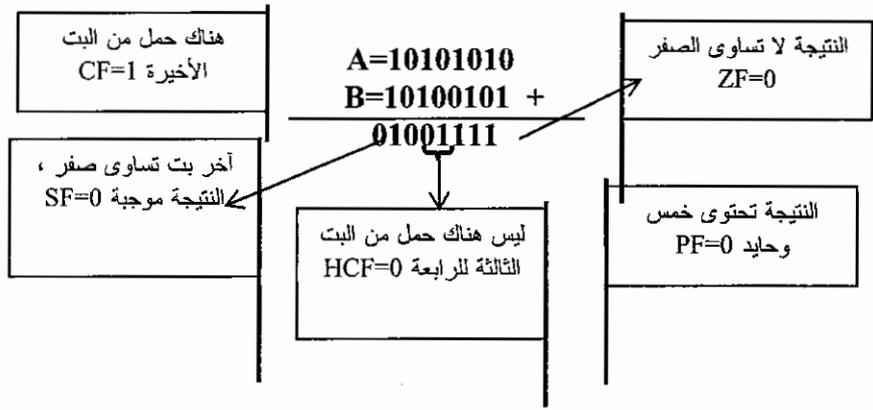
افترض المحتويات التالية للمسجلين A و B :

B	A
A5	AA

بعد تنفيذ الأمر ADD B ستصبح المحتويات كالتالي :

B	A
A5	4F

ولكى نرى كيف تتأثر الأعلام بنتيجة هذه العملية سنجرى عملية الجمع على الشفرات الثنائية لكل من الرقمين كالتالى :



الآن افترض أننا نفذنا أمر الطرح B SUB على المحتويات الأولى للمسجلين أى A=AAH و B=A5H فإنه بعد تنفيذ هذا الأمر ستصبح محتويات المسجلين كالتالى :



ولكى نرى كيف تمت عملية الطرح وكيف تأثرت الأعلام سنجرى عملية الطرح على الشفرات الثنائية لمحتويات المسجلين A و B . كما نعلم فإن عملية الطرح الثنائى يتم تحويلها إلى عملية جمع حيث سنجمع محتويات المسجل A مع المتمم الثنائى لمحتويات المسجل B (انظر الملحق الأول فى نهاية الكتاب لمراجعة عمليات الجمع والطرح الثنائى) . المتمم الثنائى لمحتويات المسجل B (10100101) هو 01011011 وبذلك تصبح عملية الطرح عملية جمع كالتالى :

$$\begin{array}{r}
 A = 10101010 \\
 + \text{المتمم الثنائى لمحتويات المسجل B } 01011011 \\
 \hline
 00000101
 \end{array}$$

بالنظر لهذه النتيجة ستكون الأعلام كالتالى :

- طالما أن البت الأولى على الأقل لا تساوى صفرا فالنتيجة لا تساوى صفرا ويكون علم الصفر يساوى صفرا أى ZF=0 .
- تحتوى النتيجة على عدد زوجى من الواحد (اثنين) ، بذلك سيكون علم الباريتى يساوى واحدا أى PF=1 .

- هناك حمل من البت الثالثة إلى البت الرابعة لذلك فعلم الحمل النصفى يكون واحدا $HCF=1$.
- آخر بت (رقم 7) تساوى صفرا لذلك فالنتيجة موجبة وعلم الإشارة يكون دائما مساويا لمحتويات آخر بت إذن $SF=0$.
- المفروض أنه فى عملية الطرح يهمننا أن نعرف إذا كان هناك استلاف أم لا لأنه فى عملية الطرح لن يكون هناك حمل . بما أن عملية الطرح قد حولت إلى عملية جمع لذلك فإنه إذا كان هناك حمل فى عملية الجمع فإن ذلك يعنى أنه لن يكون هناك استلاف فى عملية الطرح ويكون العلم $CF=0$ وهى الحالة التى نحن بصددنا الآن والعكس صحيح إذا لم يكن هناك حمل فى عملية الجمع .

2-4-4 الأمان ADI و SUI

الأمر الأول اختصار لعبارة Add Immediate أى ، اجمع المعلومة الفورية أو الثابت ، والأمر SUI اختصار لعبارة Subtract Immediate والتى تعنى أيضا اطرح المعلومة الفورية أو الثابت ، وقد أشرنا سابقا إلى أن الحرف I فى أى أمر يعنى التعامل مع معلومة فورية أو ثابت يعطى فى الأمر نفسه . الصورة العامة للأمر ADI هى :

ADI data8

$A \leftarrow A + \text{data8}$

والشفرة الست عشرية العامة لهذا الأمر هى :

C6

data8

حيث يقوم هذا الأمر بجمع محتويات البايت الثانية فى الأمر مع محتويات مسجل التراكم A ويضع النتيجة فى المسجل A وعلى ضوء هذه النتيجة تتأثر جميع الإعلام . لاحظ أن هذا الأمر يتكون دائما من اثنين بايت ، واحدة هى شفرة الأمر والثانية هى البايت أو الرقم المطلوب جمعه مع المسجل A .

الصورة العامة للأمر SUI هى :

SUI data8

$A \leftarrow A - \text{data8}$

والشفرة الست عشرية لهذا الأمر هى :

D6

data8

حيث يقوم هذا الأمر بطرح محتويات البايث الثانية في الأمر من محتويات مسجل التراكم A ويضع النتيجة في المسجل A وعلى ضوء هذه النتيجة تتأثر جميع الأعلام . لاحظ أن هذا الأمر أيضا يتكون من اثنين بايت .

مثال 10-4

المطلوب جمع الثابت أو المعلومة الفورية 05 على محتويات المسجلات B و C و D . شكل (4-5) يبين مخطط السير وشفرات الأسمبلى لهذا البرنامج . ملاحظة مهمة في هذا البرنامج هي أنه لكي نجمع الثابت مع أي مسجل فإننا نحضر أو ننقل محتويات المسجل أولا إلى مسجل التراكم A ثم نجمع الثابت مع المسجل A وبالطبع فإن النتيجة تكون في المسجل A فنقوم بنقلها إلى المسجل الآخر ثانية . وهل يوجد حل آخر!؟

3-4-4 الأوامر ADC و SBB

الأمر الأول يعنى Add with Carry أى اجمع مع الحمل ، والثانى يعنى Subtract with Borrow أى اطرح مع الاستلاف والصورة العامة للأمر ADC هي :

ADC reg

$$A \leftarrow A + CY + \text{reg}$$

حيث يقوم هذا الأمر بجمع محتويات المسجل reg مع محتويات علم الحمل CY (لاحظ أن علم الحمل يكون إما صفرا أو واحد) مع محتويات مسجل التراكم A والنتيجة توضع بالطبع في المسجل A . الشفرة الثنائية لهذا الأمر هي :

10001xxx

حيث تستبدل ال xxx بشفرة المسجل المراد التعامل معه .

الصورة العامة للأمر SBB هي :

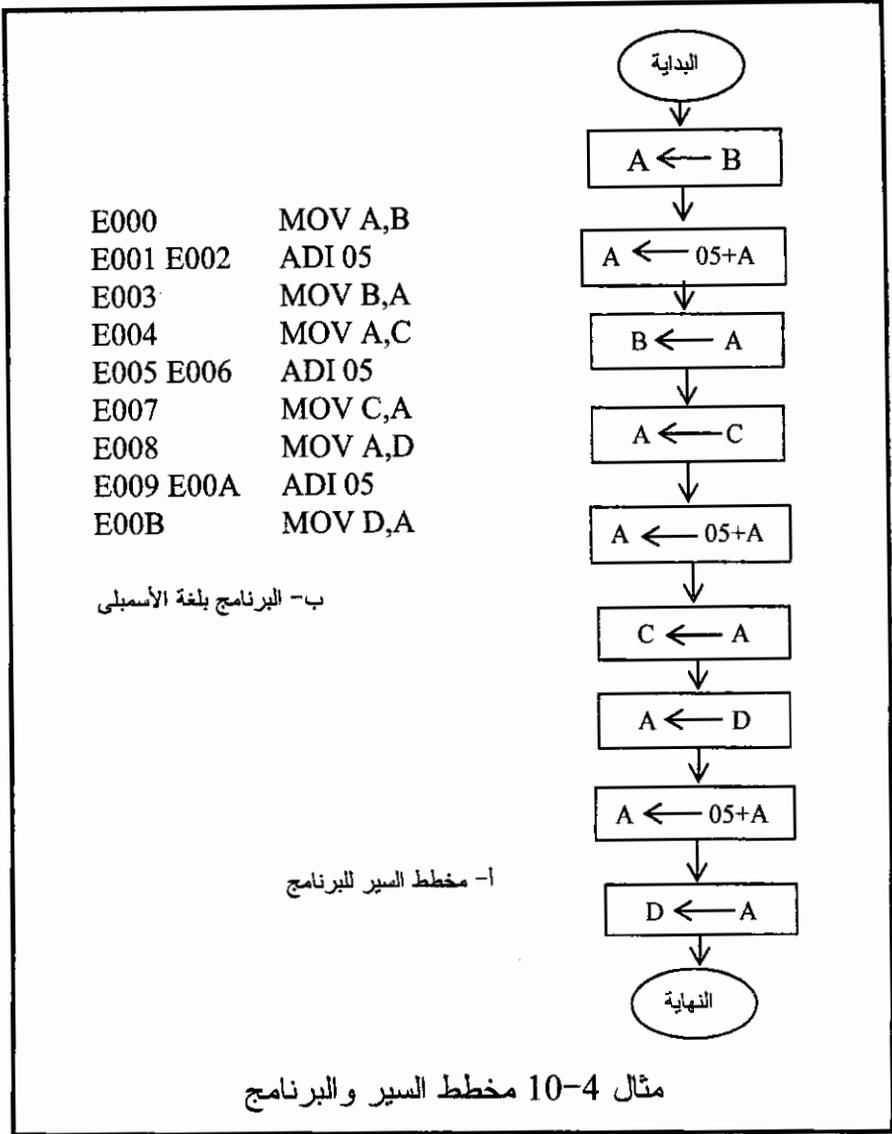
SBB reg

$$A \leftarrow A - CY - \text{reg}$$

حيث يطرح هذا الأمر من محتويات مسجل التراكم A محتويات المسجل reg ومحتويات علم الحمل CY مع ملاحظة أن المسجل A يكون هو المطروح منه دائما في جميع أوامر الطرح . الشفرة الثنائية لهذا الأمر هي :

10011xxx

حيث تستبدل ال xxx بشفرة المسجل المراد التعامل معه .



مثال 11-4

المطلوب جمع الرقمين 9A35H و 23F9H ووضع نتيجة الجمع في أماكن الذاكرة E100 و E101 و E102. المشكلة في هذين الرقمين أن كلا منهما يشغل اثنين بايت) ونحن نعلم أن جميع أوامر الجمع تتعامل مع بايت واحدة فقط فما الحل؟ الحل لهذه المشكلة ممكن بأن نضع الرقم الأول مثلا في المسجلين B و C بحيث يحتوى المسجل B البايت ذات القيمة العظمى من الرقم وهى 23

ويحتوى المسجل C البايت ذات القيمة الصغرى وهى F9 وبنفس الطريقة نضع الرقم الثانى فى المسجلين D و E بحيث يحتوى المسجل D البايت 9A ويحتوى المسجل E البايت 35 . بعد ذلك سنجمع البايت ذات القيمة الصغرى فى كل من الرقمين أى المسجل C زائد المسجل E باستخدام الأمر ADD وسينتج عن ذلك نتيجة وحمل ، النتيجة نخزنها فى الذاكرة E100 ، وبعد ذلك نستخدم الأمر ADC لجمع البايت ذات القيمة العظمى فى الرقمين مع محتويات علم الحمل وهى الحمل الناتج من عملية الجمع الماضية وسينتج عن ذلك نتيجة تخزن فى الذاكرة E101 وحمل ، هذا الحمل يخزن فى الذاكرة E102 كجزء من النتيجة . شكل (4-6) يبين رسماً توضيحياً للحل وخريطة التدفق والبرنامج لهذا المثال .

قبل أن نترك الأمر ADC يجب أن نفهم جيداً متى يكون من الضرورى استخدام الأمر ADC ؟ ومتى يكون من الضرورى عدم استخدامه ؟ فمثلاً فى المثال السابق (4-11) كان من الضرورى عدم استخدام الأمر ADC فى عملية الجمع الأولى (E + C) ولكن يجب استخدام الأمر ADD خوفاً من أن يكون علم الحمل CY به واحد من أى عملية سابقة ونحن لا ندرى فيجمع مع عملية الجمع الأولى وتكون النتيجة خاطئة . أما فى عملية الجمع الثانية (D + B + CY) فإنه لا بد من استخدام الأمر ADC لأننا نريد أن نأخذ قيمة علم الحمل CY فى الاعتبار .

4-4-4 الأمان INR و DCR

الأمر الأول يعنى Increment أى زد المحتويات بمقدار واحد ، و Decrement أى انقص المحتويات بمقدار واحد ، والصورة العامة للأمر INR هى :

INR reg

reg ← 1 + reg

حيث يقوم هذا الأمر بجمع واحد على محتويات المسجل reg أو مكان الذاكرة M الذى عنوانه فى HL . الصورة الثنائية لهذا الأمر هى :

00xxx100

حيث xxx تستبدل بشفرة المسجل المراد التعامل معه .
الصورة العامة للأمر DCR هى :

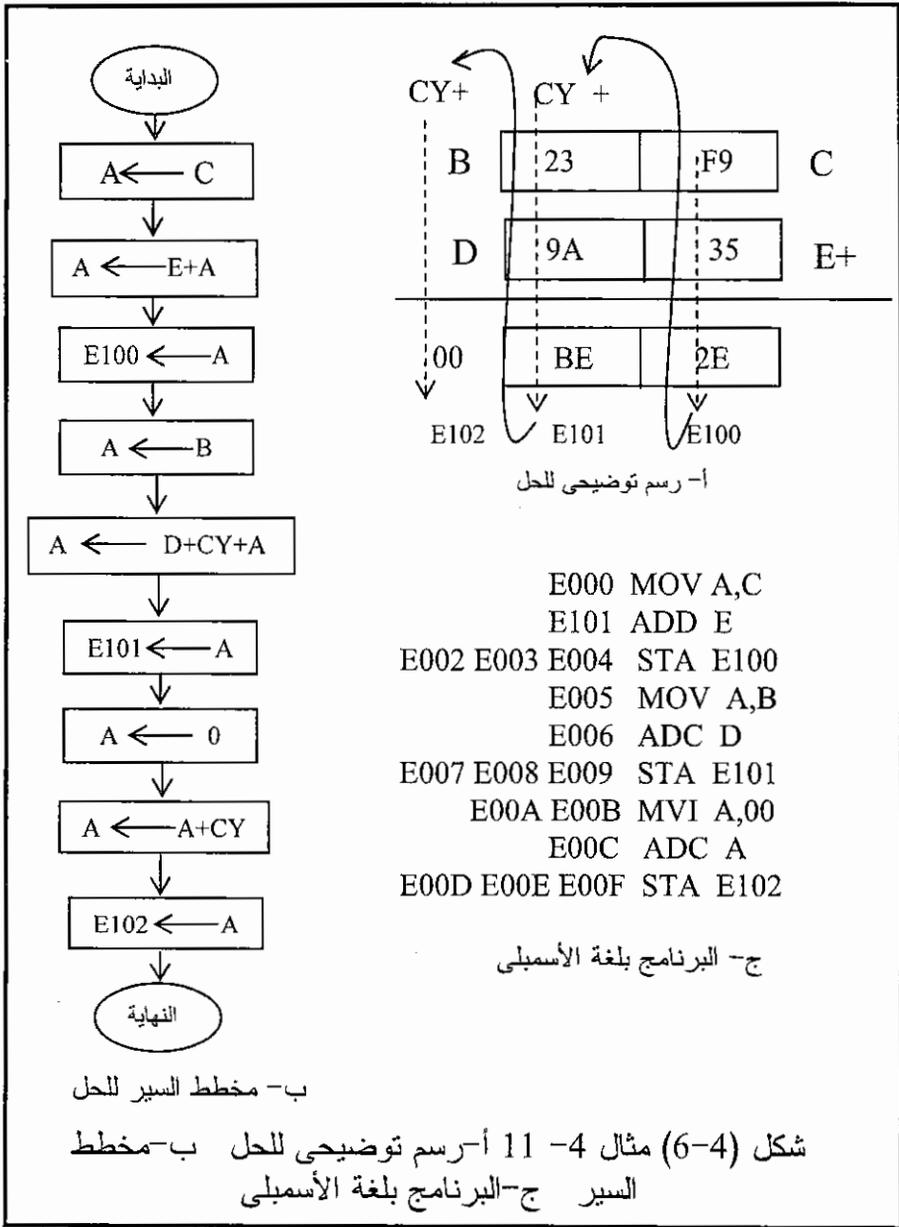
DCR reg

reg ← reg - 1

حيث يقوم هذا الأمر بطرح واحد من محتويات المسجل reg أو مكان الذاكرة M الذى عنوانه فى HL . الصورة الثنائية لهذا الأمر هى :

00xxx101

حيث xxx تستبدل بشفرة المسجل المراد التعامل معه .



ب- مخطط السير للحل

شكل (4-6) مثال 4-11 أ-رسم توضيحي للحل ب-مخطط السير ج-البرنامج بلغة الأسمبلى

5-4-4 الأوامر INX و DCX

الأوامر INX و DCX هما مرادفتان للأمرين INR و DCR ولكنهما يستخدمان مع أزواج مسجلات ، فالأمر INX يجمع واحدا على محتويات زوج من المسجلات والصورة العامة الأسمبلى والثنائية له هي :

INX rp

00xx0011

rp ← 1 + rp

حيث xx تستبدل بشفرة زوج المسجلات المراد التعامل معه . وأما الأمر DCX فإنه يطرح واحدا من محتويات زوج من المسجلات وصورته العامة (الأسمبلى والثنائية) هي :

DCX rp

00xx1011

rp ← rp - 1

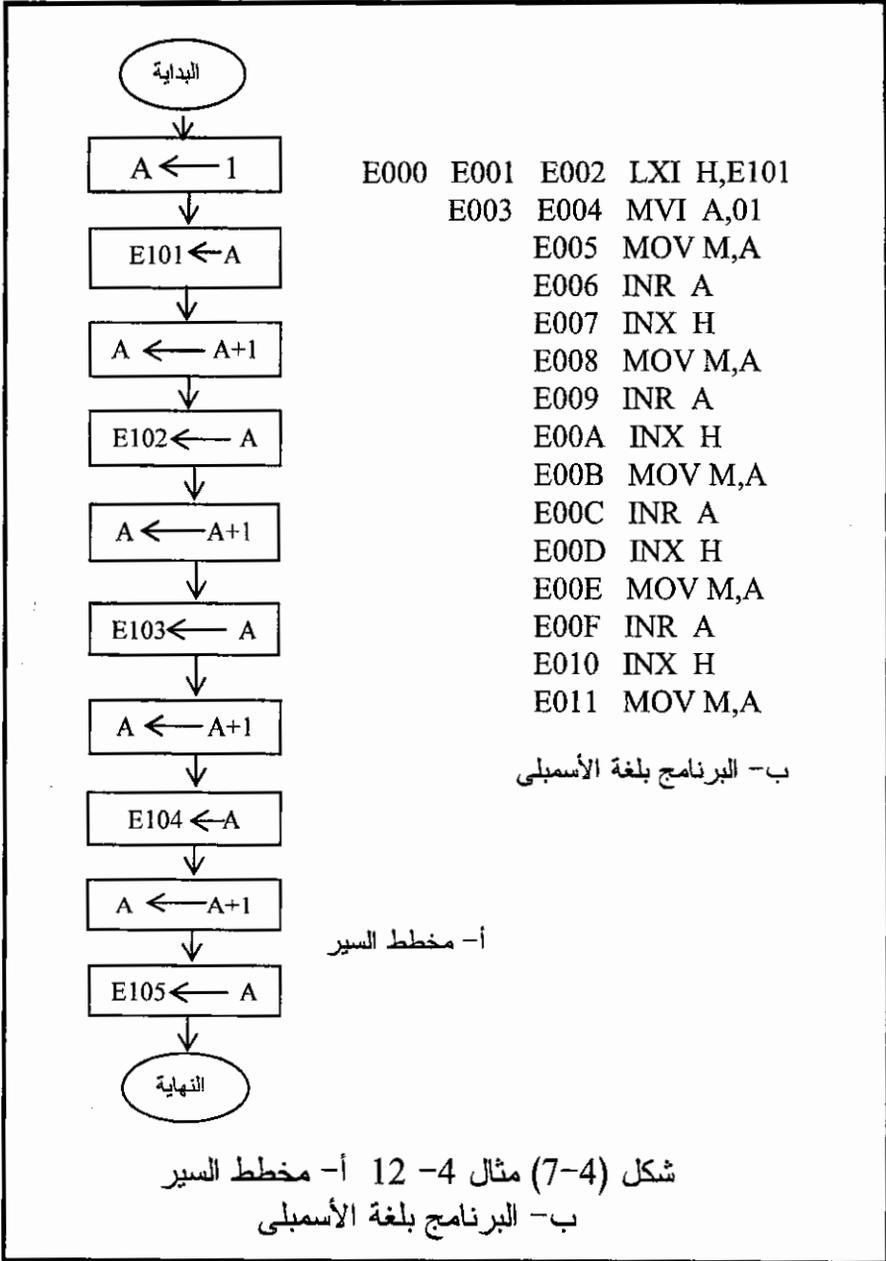
حيث تستبدل xx بشفرة زوج المسجلات المراد التعامل معه .

مثال 4-12

المطلوب تخزين الأرقام 05, 04, 03, 02, 01 في أماكن الذاكرة التي عناوينها E105, E104, E103, E102, E101 على التوالي . شكل (4-7) يبين خريطة التدفق والبرنامج لحل هذه المسألة . لاحظ استخدام الأمر INR لزيادة واحد على محتويات المسجل A والأمر INX لزيادة واحد على محتويات زوج المسجلات HL في هذا المثال . نلاحظ من شكل (4-7) أننا لكي نخزن خمسة أرقام في الذاكرة كتبنا برنامجا مكونا من ثمانية عشرة بايت (E000 حتى E011) فما العمل لو أننا نريد تخزين ألف رقم أو أكثر ، كم ستشغل من ذاكرة للبرنامج...! هل هناك من حل لتخفيض عدد أوامر مثل هذه البرامج ؟ إن الحل لذلك هو استخدام أوامر القفز والحلقات وهو موضوع الجزء القادم بعد أن نعرض بعض التمارين كتطبيق على أوامر الحساب .

4-5 تمارين

1. اكتب برنامجا يجمع محتويات أماكن الذاكرة E104, E103, E102, E101 مع ما يناظرها من الأماكن E10E, E10D, E10C, E10B, E10A ثم يضع النتيجة في الأماكن E114, E113, E112, E111, E110 .
2. أعد البرنامج السابق مستخدما الطرح بدلا من الجمع .
3. اكتب برنامجا يجمع الرقم F3A56BH مع الرقم 78B6A9H ويضع النتيجة في بايتات الذاكرة E103, E102, E101, E100 .
4. أعد البرنامج السابق مستخدما الطرح بدلا من الجمع .



4-6 مجموعة أوامر القفز

Jump Instructions

القاعدة العامة أن المعالج يقوم بتنفيذ البرنامج حسب ترتيب الأوامر الموجودة فيه من أول البرنامج إلى نهايته ، ولقد كنا حريصين في جميع الأمثلة السابقة على الحفاظ على هذه القاعدة ، ولكن هناك بعض التطبيقات التي تتطلب الخروج على هذه القاعدة كأن يطلب منك مثلا تنفيذ عملية معينة عدد معين من المرات أو عدد لا نهائي من المرات . فعندما يكون المعالج مثلا مراقبا لدرجة الحرارة في عملية صناعية معينة فإن عليه أن يقرأ درجة الحرارة ويقارنها بدرجة حرارة مخزنة في الذاكرة كمرجع ، وإذا زادت الحرارة عن حد معين يقوم المعالج بضرب جرس إنذار مثلا وإذا نقصت عن حد معين يشغل سخان لزيادتها ، مثل هذا البرنامج سيكون عبارة عن مجموعة من الأوامر التي تنفذ إلى مالانهاية طالما أن المعالج يراقب درجة الحرارة .

لقد أتاح المعالج هذه العملية بتوفير بعض الأوامر التي تمكنك كمبرمج من القفز بعملية التنفيذ من مكان لآخر خلال البرنامج . هناك نوعان من القفز :

4-6-1 القفز غير المشروط unconditional jump

عند تنفيذ أي عملية قفز غير مشروط ينتقل المعالج بعملية التنفيذ إلى المكان الجديد دون أي قيد أو شرط ، وهناك أمر واحد فقط من أوامر المعالج 8085 يقوم بهذه العملية ، والصورة العامة لهذا الأمر هي :

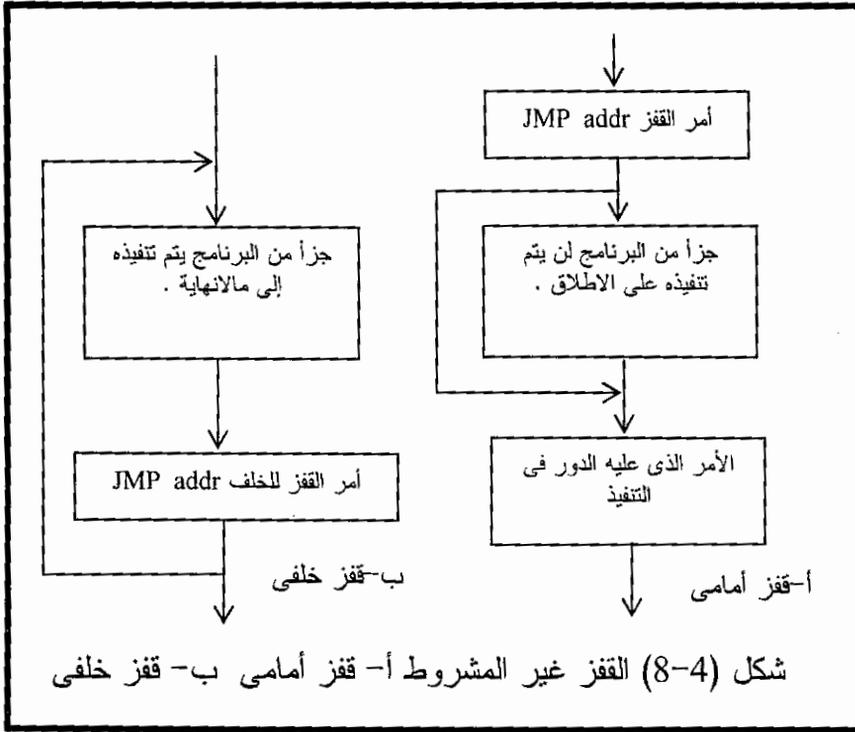
JMP addr

حيث إنه عند تنفيذ هذا الأمر يوضع العنوان addr الذي سيتم القفز إليه في عداد البرنامج فيصبح الأمر الموجود عند هذا العنوان هو الأمر الذي عليه الدور في التنفيذ . لاحظ أن هذا الأمر يتكون من ثلاث بايتات واحدة هي شفرة الأمر واثنان للعنوان addr الذي سيتم القفز إليه .

إن القفز باستخدام الأمر JMP addr قد يكون إلى الأمام في البرنامج وقد يكون إلى الخلف . إذا كان القفز إلى الأمام سينتج عن ذلك وجود جزء من البرنامج لن ينفذ على الإطلاق وهو الجزء الذي بين أمر القفز JMP والأمر الذي سيتم القفز إليه . أما إذا كان القفز إلى الخلف فإنه سينتج عن ذلك ما يسمى بالحلقة اللانهائية Infinite loop والتي سوف يستمر المعالج في تنفيذها إلى مالانهاية . شكل (4-8) يبين مخطط السير لعملية القفز غير المشروط الأمامي والخلفي .

4-6-2 القفز المشروط Conditional jump

كما يوحي الاسم فإنه في هذا النوع من القفز لن يتم القفز إلا إذا تحقق شرط معين أما إذا لم يتحقق الشرط فإن البرنامج ينفذ في المتتابع الطبيعي حيث سينفذ الأمر الذي بعد أمر القفز مباشرة . إن شروط القفز توضع دائما على الأعلام التي في مسجل الحالة ، فيمكنك مثلا أن تجعل القفز مشروطا بأن تكون النتيجة صفرا أو تجعل القفز مشروطا بأن تكون النتيجة سالبة ، وهكذا حيث أن هناك خمسة أعلام واحد منها وهو علم الحمل النصفى HC لا يستخدم كشرط في عمليات القفز فإنه يتبقى أربعة أعلام يمكن أن تستخدم في ثمانية من أوامر القفز المشروط كالتالي :



- أقفز إذا كانت النتيجة صفرا JZ addr
- أقفز إذا كانت النتيجة ليست صفرا JNZ addr
- أقفز إذا كانت النتيجة سالبة JM addr
- أقفز إذا كانت النتيجة موجبة JP addr
- أقفز إذا كان هناك حمل JC addr
- أقفز إذا لم يكن هناك حمل JNC addr
- أقفز إذا كانت الباريتمى فردية JPO addr

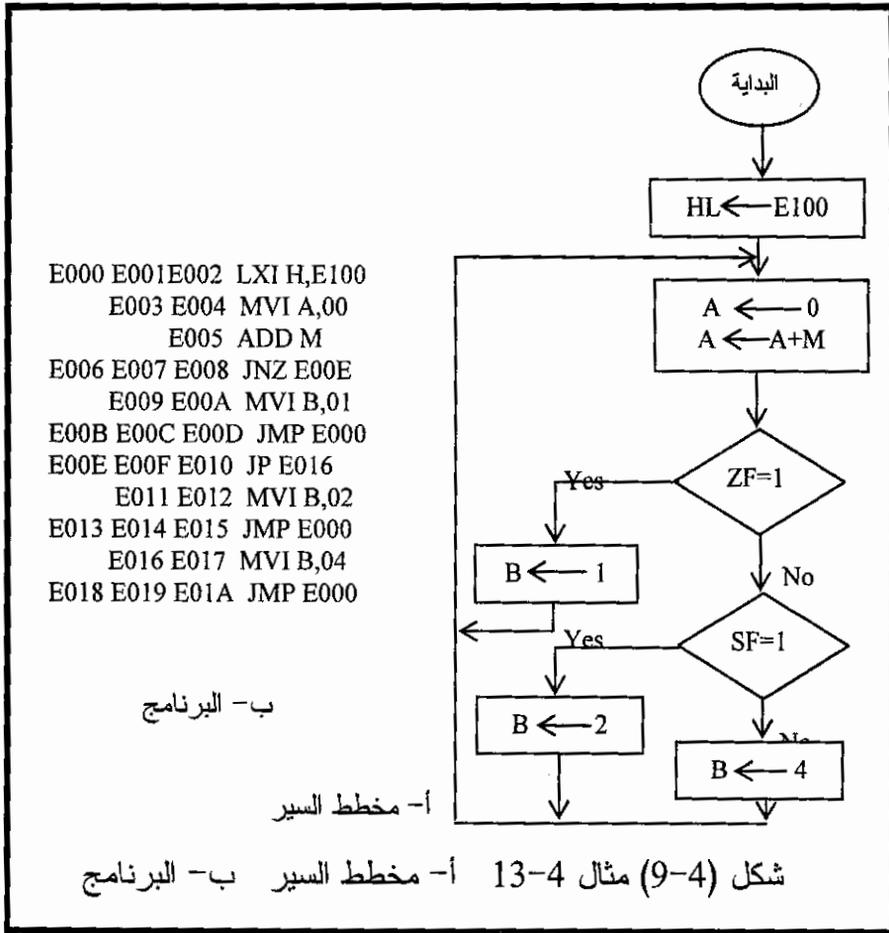
لاحظ أن عدد هذه الأوامر ثمانية ، إثتان منها لكل علم من الأعلام الأربعة تمثل جميع الحالات التى يمكن أن يكون فيها هذا العلم (صفرا أو واحدا) . إن جميع هذه الأوامر لا بد وأن تكون ثلاث بايتات ، واحدة هى شفرة الأمر op code وإثنتان للعنوان الذى سيتم القفز إليه إذا تحقق الشرط . لاحظ أن النتيجة التى نعتبها فى الأوامر السابقة هى آخر نتيجة تأثرت بها الأعلام ، ولذلك فإنه قيل أن نكتب أى أمر من أوامر القفز غير المشروط يجب أن ندرس جيدا هل الأمر السابق لأمر القفز المشروط يؤثر على الأعلام أم لا كما سيتضح من المثال التالى :

مثال 4-13

اكتب برنامجا يقرأ محتويات بايت الذاكرة E100 باستمرار (إلى مالانهاية) ثم يختبر هذه المحتويات بحيث إذا كانت صفرا يضع 1 فى المسجل B وإذا كانت سالبة يضع 2 فى المسجل B وإذا كانت موجبة يضع 4 فى نفس المسجل B . شكل (4-9) يبين خريطة التدفق والبرنامج لهذا المثال . البرنامج الموجود فى شكل (4-9) يعتبر تدريبا على معظم أفكار الحلقات ، ففيه الحلقات اللانهائية الناتجة عن الأمر JMP addr ، كما أن فيه القفز المشروط JNZ addr ، JP addr كما أن فيه القفز إلى الأمام فى البرنامج ، والقفز إلى الخلف أيضا لذلك يجب دراسة هذا البرنامج بدقة وإمعان . هناك أوامر أخرى يتسبب عنها قفز بعملية تنفيذ البرنامج إلى أماكن أخرى وهى أوامر القفز إلى البرامج الفرعية والعودة منها وسوف نرجىء الدراسة التفصيلية للبرامج الفرعية إلى فصل آخر خاص بذلك .

4-7 مهمة أخرى للأسمبلر

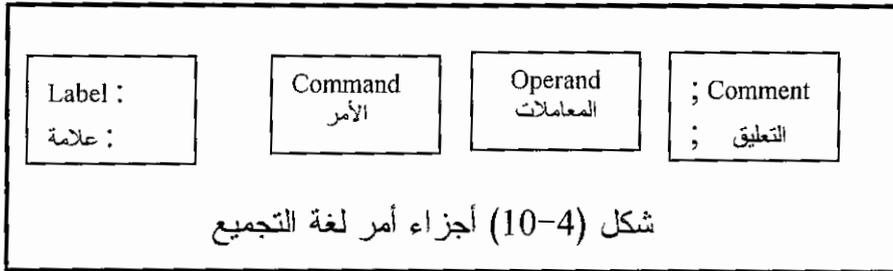
المهمة الوحيدة التى عرفناها حتى الآن للأسمبلر هى مهمة تحويل الشفرات الحرفية (الأسمبلى) إلى شفرات ثنائية أو لغة الماكينة ، ولكن لحسن الحظ فإن الأسمبلر قد تم تزويده ببعض المهام الأخرى التى تريح المبرمج إلى درجة كبيرة والتى سنتعرف على واحدة منها فى هذا الجزء . ينظر أى أسمبلر إلى الأمر الذى تكتبه له على أنه مكون من أجزاء أربعة كالموضحة فى شكل (4-10) والتى سندرسها بالترتيب فيما يلى :



1-7-4 الأوامر والمعاملات

لقد درسنا الكثير من أوامر لغة الأسمبلى والتي سنلخصها كلها فى شكل واحد وفى ترتيب أبجدى فى آخر هذا الفصل إن شاء الله . من هذه الأوامر MOV, ADD, JMP, وغيرها . كل أمر من هذه الأوامر لا بد وأن يكون له معاملات وهذه المعاملات هى المسجلات أو أماكن الذاكرة التى سوف يؤثر عليها الأمر ، فمثلا الأمر MOV A,B معاملاته هى المسجلين A, B والأمر ADD C معاملاته هى المسجل C والأمر ADD M معاملاته هى بايت الذاكرة التى عنوانها فى المسجلين HL وهكذا . هناك أوامر قليلة ليست لها معاملات على الإطلاق ومنها الأمر NOP والذى معناه لا تعمل شيئا No Operation كما أن بعض المعاملات فى أى أمر لا يزيد عن اثنين بأى حال . إن الأمر يجب أن يفصل عن معاملاته بمسافة واحدة على الأقل وإن زاد عدد المسافات عن واحدة فلن يضر ذلك فى شيء حيث أن الأسمبلى يهملها ، أما إذا لم توجد مسافة واحدة على

الأقل بين الأمر ومعاملاته فإن الأسمبلر سيعطى خطأ على ذلك ولن يقبل منك هذا الأمر . إذا كان الأمر له معاملان كالأمر MOV A,B فإن المعاملين يجب أن يفصل بينهما بفاصلة (,) وإذا لم يحدث ذلك فإن الأسمبلر يعطى رسالة خطأ على ذلك .



2-7-4 التعليق Comment

حتى تجعل برنامجك مفهوماً ومن السهل قراءته وتتبعه بالنسبة للآخرين فإنه يجب عليك أن تكتب بعض التعليقات البسيطة بجانب كل أمر . لذلك فإن الأسمبلر يعطيك فرصة أن تكتب أى شيء تريده فى نهاية الأمر على أن تفصل بين الأمر والتعليق بفاصلة منقوطة يفهم منها الأسمبلر أن كل المكتوب بعدها يعتبر تعليقا ولا يدخل ضمن الأمر . إذا كان التعليق الذى ستكتبه سيأخذ أكثر من خط واحد فإن كل خط يجب أن يبدأ بفاصلة منقوطة . المثال التالى سيوضح أهمية كتابة التعليق فى نهاية الأمر .

مثال 4 . 14

```

; Multiplication of two numbers
E000 E001 MVI A,00 ; Put 0 into A, Clean accumulator
          E002 ADD B      ; Addition of A and B, result goes to A
E003 DCR C      ; Decrement C by 1
E004 E005 E006 JNZ E002 ;Go to add B to itself one more time
                      ;as long as C not equal to 0
E007 E008 E009 STA E100 ; store result in location E100
;-----

```

هذا البرنامج يضرب الرقم الموجود فى المسجل B فى الرقم الموجود فى المسجل C ويضع النتيجة فى بايت الذاكرة رقم E100 . بما أنه ليس هناك أمر من أوامر الشريحة 8085 يقوم بإجراء عملية الضرب لذلك كان لابد أن نكتب هذا البرنامج لإجراء عملية الضرب عن طريق تكرار الجمع حيث يجمع البرنامج محتويات المسجل B مع نفسه عدد من المرات يساوى الرقم الموجود

في المسجل C . ستجد أن التعليقات المكتوبة بجانب الأوامر تدل على ذلك .
التعليق الموجود في السطر الأول مثلا يقول "ضع صفرا في A ، تنظيف المركم
" وهذا ضروري حتى لا نجمع أى رقم قد يكون في المسجل A قبل البدء في
عملية جمع المسجل B مع نفسه . التعليق في السطر الثاني يقول "جمع A+B ،
النتيجة في A " لاحظ أن هذا الأمر هو بداية حلقة يتم القفز إليها من السطر
الرابع وفي المرة الأولى من دخول هذه الحلقة سيجمع المعالج المسجل A=0 مع
محتويات المسجل B فيصبح A=B . التعليق في السطر الثالث يقول "انقص
محتويات C بمقدار واحد" . التعليق في السطر الرابع يقول "ارجع لجمع المسجل
B مع نفسه مرة أخرى طالما أن C لا يساوى صفرا ، النتيجة في A . التعليق
في السطر الخامس يقول " خزن النتيجة في البايث E100" . التعليق في السطر
السادس وهو الخط المنقط تم وضعه فقط للدلالة على نهاية البرنامج وهذا أحيانا
يكون مهما جدا في عملية تنظيم كتابة البرنامج خاصة إذا كان البرنامج مكونا
من أكثر من جزء حيث يمكن الفصل بين الأجزاء المختلفة عن طريق مثل هذه
الخطوط التي يعرفها الأسمبلر على أنها تعليقات . بالطبع يجب أن تكون
التعليقات باللغة الإنجليزية ، إلا إذا كان محرر الكلمات الذى تستخدمه في كتابة
البرنامج سيسمح لك باستخدام اللغة العربية في التعليقات .

4-7-3 العلامة Label

في جميع البرامج التي كتبناها حتى الآن كنا حريصين تماما على أن نكتب
عناوين البايثات التي يشغلها أى أمر من أوامر البرنامج ، وكانت هذه العملية
ضرورية بالذات عند التعامل مع أوامر القفز التي نحتاج فيها لمعرفة عنوان
الأمر الذى سنقفز إليه مثل الأمر الرابع في المثال السابق (مثال ضرب الرقمين
4-14) وهو JNZ E002 ، فلولا أننا كنا نكتب عناوين البايثات التي عندها
الأوامر لما حددنا أن القفز يجب أن يكون إلى الأمر الثاني E002 ADD B . في
الحقيقة إن مهمة تتبع العناوين لجميع أوامر البرنامج تعتبر مهمة صعبة للمبرمج
، بالذات إذا كان البرنامج يحتوى على الكثير من أوامر القفز ، وتعتبر مهمة
سهلة للأسمبلر يستطيع القيام بها في نفس الوقت . فطالما أن الأسمبلر يعرف
جيدا عدد البايثات التي يتكون منها كل أمر فلم لا يتولى هو عملية العنونة
لأوامر البرنامج على أن يعطيه المبرمج فقط عنوان أول أمر في البرنامج ، وما
على المبرمج في هذه الحالة إلا الكتابة فقط بالشفرات الأسمبلى ، هنا يبرز سؤال
مهم : كيف سنحدد للأسمبلر الأوامر التي من الممكن أن يتم القفز إليها ؟ إن
ذلك يتم عن طريق العلامات Labels التي نضعها قبل الأمر المراد القفز إليه
على أن نستخدم نفس العلامة في أمر القفز نفسه ، إننا سنعيد كتابة البرنامج

الموجود في المثال 4-14 مرة أخرى دون استخدام عناوين للأوامر وباستخدام العلامات كما في المثال 4-15 .

مثال 4-15

```
MVI A,00 ; Put 0 into accumulator
HERE: ADD B ; A+B into A
      DCR C ; Decrement C by 1
      JNZ HERE ; Jump to add B to itself one more
           ; time as long as C not equal to 0
      STA E100 ; Store result in E100
```

لاحظ أننا وضعنا العلامة وهي كلمة: HERE كعلامة عند الأمر الذي سيتم القفز إليه وهو الأمر الثاني في البرنامج السابق ، واستخدمنا نفس العلامة HERE في الأمر JNZ HERE كي يتم القفز إلى الأمر رقم 2 . من البديهي أنه لا بد وأن يكون هناك تطابق تام بين العلامة في الأمر الذي سيتم القفز إليه والعلامة في أمر القفز نفسه بحيث إذا لم يوجد هذا التطابق فإن الأسمبرل سيعطى خطأ . أيضا لا بد وأن تنتهي العلامة الموجودة في أول الأمر الذي سيتم القفز إليه بالحرف (:). ليميز بها الأسمبرل بين نهاية العلامة وبداية الأمر كما أن عدد أحرف العلامة يجب ألا يزيد عن ثمانية أحرف لا تبدأ برقم .

البرنامج الموجود في المثال 4-14 مكتوب في الذاكرة ابتداء من البايث E000 ، افترض مثلا أننا لأي سبب نريد نقل البرنامج بحيث يبدأ من البايث E050 بدلا من E000 . في هذه الحالة لا بد وأن نعيد فحص البرنامج بدقة ونغير العناوين الموجودة في جميع أوامر القفز في البرنامج ، فمثلا الأمر JNZ E002 في المثال 4-14 سيصبح JNZ E052 لو أننا بدأنا البرنامج من البايث E050 ، وهكذا تخيل لو أن البرنامج به العديد من أوامر القفز فإنه لا شك ستكون عملية إعادة عنونة الأوامر من الصعوبة بمكان وبالذات في البرامج الطويلة . إن استخدام العلامات Labels كما في مثال 4-15 يريح المبرمج تماما من عملية تتبع العناوين في أوامر القفز حتى لو تغير مكان البرنامج لأنك تعطى الأسمبرل عنوان بداية البرنامج فقط وهو ينسب جميع العلامات إلى عنوان البداية وهذه في الحقيقة فائدة عظيمة يوفرها الأسمبرل للمبرمج . لذلك فإن جميع البرامج التي سنعرضها فيما بعد سنكتبها باستخدام لغة الأسمبرل والعلامات دون الحرص على كتابة عناوين الأوامر لأننا بذلك نكون قد عبرنا مرحلة لا بأس بها في لغة الأسمبرل .

8-4 أوامر الإدخال والإخراج Input Output instructions

إلى الآن رأينا كيف نبرمج شريحة المعالج وكيف نحرك المعلومات داخلها من مسجل إلى مسجل آخر أو من مسجل إلى الذاكرة أو العكس ولكن لم نعرف حتى الآن كيف نظهر معلومة على شاشة عرض مثلا أيا كان نوع هذه الشاشة ، أو كيف ندخل معلومة إلى المعالج من خلال لوحة مفاتيح على سبيل المثال . إن شاشة العرض ولوحة المفاتيح يعتبران مثالان من ضمن العديد من الأمثلة التي تحتاج لعمليات إخراج وإدخال البيانات . فمثلا حينما يستخدم المعالج فى التحكم فى أى متغير فى أى عملية صناعية وليكن مثلا درجة الحرارة فإنه لابد من إدخال درجة الحرارة إلى المعالج بعد تهيئتها ووضعها فى الصورة المناسبة ، وكذلك إذا أراد المعالج رفع درجة الحرارة أو ضرب جرس إنذار فإنه لابد وأن يخرج إشارة معينة تؤخذ وتهدأ فى الصورة المناسبة للجهاز الذى ستذهب إليه سواء كان سخانا أو جرسا أو غيره . جميع عمليات الإدخال والإخراج تتم من خلال ما يسمى ببوابات الإدخال والإخراج والتعامل مع هذه البوابات دائما ينقسم إلى قسمين : قسم خاص بالبناء الإلكتروني لهذه البوابات وكيفية توصيلها مع المعالج وهذا القسم سندرسه بالتفصيل فى فصل قادم إن شاء الله ، والقسم الآخر هو كيفية برمجة المعالج للتعامل مع هذه البوابات وهو موضوع دراستنا فى هذا الجزء حيث سندرس الأوامر الخاصة بهذه العملية وسنفترض فى دراستنا فى هذا الجزء أن القارئ لديه على الأقل بوابة إدخال واحدة وبوابة إخراج واحدة موصلان على الميكروكومبيوتر الذى يتدرب عليه ويكتب عليه برامجه .

1-8-4 أمر الإدخال IN وأمر الإخراج OUT

وهما اختصار لكلمتى Input يعنى أدخل و Output يعنى أخرج والصورة العامة للأمر IN هي :

IN no.

البوابة رقم no. ← المسجل A

حيث سيقوم هذا الأمر بإدخال المعلومة الموجودة على بوابة الإدخال والتي رقمها no. فى الأمر نفسه إلى مسجل التراكم A . هذا الأمر يتكون من اثنين بايت واحدة هي شفرة الأمر IN والثانية هي رقم البوابة المراد التعامل معها ، ولذلك فإنه يمكن التعامل مع $2^8 = 256$ بوابة إدخال تبدأ من البوابة رقم 00H إلى البوابة رقم FFH . انظر شفرة الأمر IN الست عشرية فى جدول الأوامر فى نهاية هذا الفصل .

الصورة العامة لأمر الإخراج OUT هي :

OUT no.

المسجل A البوابية رقم no.

حيث سيقوم هذا الأمر بإخراج المعلومة الموجودة في مسجل التراكم A إلى بوابة الإخراج التي رقمها no. . هذا الأمر يتكون من اثنين بايت ، واحدة هي شفرة الأمر OUT والثانية هي رقم البوابة المراد التعامل معها ، ولذلك فإنه يمكن التعامل مع $2^8 = 256$ بوابة إخراج تبدأ أيضا من البوابة رقم 00H إلى البوابة رقم FFH . انظر شفرة الأمر OUT الست عشرية في جدول الأوامر في نهاية هذا الفصل .

لاحظ أن كلا من بوابة الإدخال وبوابة الإخراج تتكون من 8 بتات مثلها في ذلك مثل أي مسجل يتكون من 8 بتات ولذلك فإن أكبر رقم يمكن أن يكون على أي بوابة هو الرقم 255 . تذكر أيضا أن أي تعامل مع أي بوابة باستخدام هذين الأمرين لابد وأن يكون من خلال المرمك A ، فإذا أردت أن تخرج محتويات المسجل C مثلا إلى بوابة الإخراج رقم 05 فإنه لابد وأن تنقل هذه المحتويات إلى المسجل A أولا ثم تنفذ أمر الإخراج كما يلي :

انقل محتويات C إلى A ; MOV A,C

أخرج على بوابة الإخراج رقم 05 ; OUT 05

بعد تنفيذ الأمرين السابقين ستري محتويات المسجل C على بوابة الإخراج رقم 05 . أما إذا أردنا أن ندخل محتويات بوابة الإدخال رقم 00 ونخزنها في بايت الذاكرة رقم E100 فإننا ننفذ الأمرين التاليين على سبيل المثال :

أدخل محتويات بوابة الإدخال رقم 00 إلى المرمك ; IN 00

خزن محتويات المرمك في الباييت E100 ; STA E100

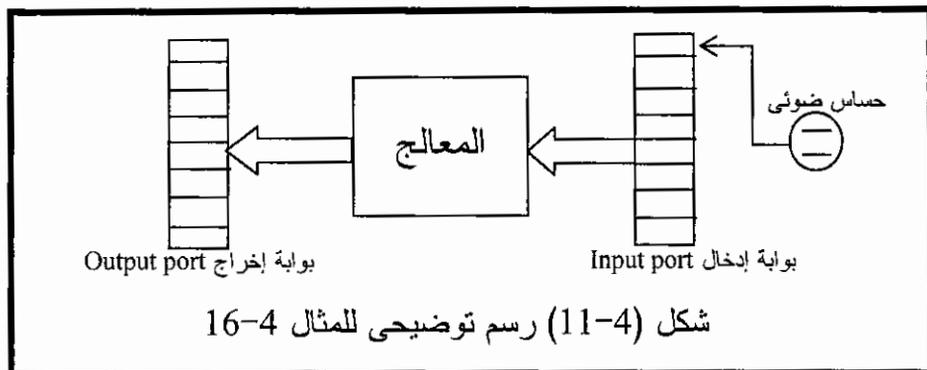
بعد تنفيذ هذين الأمرين فإنه إذا كانت محتويات بوابة الإدخال رقم 00 تساوي 55 مثلا فإن هذا الرقم (55) ستجده في الباييت E100 .

مثال 4-16

إفترض أن لدينا خط إنتاج في أحد المصانع تعبر عليه المنتجات وفي أثناء العبور فإن كل منتج يقطع خلية ضوئية فتعطي نبضة كهربية على خرجها . خرج هذه الخلية موصل على البت رقم 0 في بوابة الإدخال رقم 00H . المطلوب هو كتابة برنامج يعد هذه المنتجات ويخرج العدد على بوابة الإخراج رقم 00H أيضا . شكل (4-11) يبين رسما توضيحيا لهذه العملية وشكل (4-12) يبين مخطط السير والبرنامج لهذا المثال .

لاحظ وجود الأمر CPI وهو أحد أوامر الحساب التي لم نشرحها وتركناها للقارئ لمراجعتها من قوائم الأوامر في نهاية الفصل ، ولقد استخدم هذا الأمر

لمقارنة محتويات المسجل A القادمة من بوابة الإدخال بالقيمة 00 أولاً ، وطالما أن هذه المحتويات تساوى 00 فإن ذلك يعنى عدم مرور أى منتج على خط الإنتاج ويقوم المعالج بتكرار عملية القراءة حيث يقفز إلى HERE1: كما فى البرنامج .



عندما يمر أى منتج ستكون محتويات بوابة الإدخال مختلفة عن الصفر ولن يرجع المعالج إلى العلامة: HERE1: ثانية ولكنه سيزيد محتويات المسجل B بمقدار واحد حيث B تعتبر عدداً للمنتجات المارة على الخط ، ثم بعد ذلك سيقوم المعالج بقراءة بوابة الإدخال مرة أخرى للتأكد من أن المنتج قد مر من على الخط وذلك بمقارنة محتويات بوابة الإدخال بالقيمة 01 وطالما أنها تساوى 01 يقفز إلى العلامة: HERE2: حيث لا يعمل شيئاً سوى قراءة البوابة . عندما تنزل البوابة إلى 00 مرة أخرى فإن ذلك يدل على أن المنتج قد مر على الخط فيقوم المعالج بإخراج قيمة العداد B على بوابة الإخراج ويذهب إلى العلامة: HERE1: مرة أخرى لمواصلة البرنامج .

9-4 مجموعة أوامر المنطق Logic Instruction Set

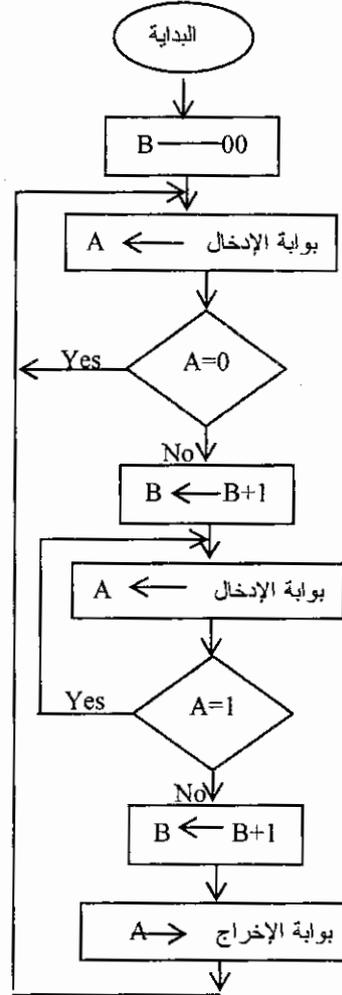
العمليات المنطقية التى يستطيع المعالج 8085 القيام بها هى العمليات AND و OR و XOR و NOT وسنكتفى هنا بعرض الصورة العامة لهذه الأوامر على أن يقوم القارئ بمراجعة هذه الأوامر فى القوائم الموجودة فى آخر الفصل لمعرفة الشفرات الست عشرية والزمن الذى يأخذه كل أمر لكى يتم تنفيذه . كما ذكرنا سابقاً فإن العمليات المنطقية مثلها مثل العمليات الحسابية لابد وأن يكون المرمك طرفاً

فيها كما أن النتيجة لا بد وأن توضع في المرمك أيضا . جميع العمليات المنطقية تؤثر على الأعلام ، انظر قوائم الأوامر في آخر الفصل لترى ذلك ولترى متى تكون هذه الأوامر مكونة من بايت واحدة ومتى تكون مكونة من 2 بايت . الصورة العامة للأمر AND هي :

```

MVI B,00
HERE1: IN 00
      CPI 00
      JZ HERE1
      INR B
HERE2: IN 00
      CPI 01
      JZ HERE2
      MOV A,B
      OUT 00
      JMP HERE1

```



شكل (4-12) مخطط السير والبرنامج للمثال 4-16

ANA reg

AND reg ← مسجل A

حيث سيقوم المعالج بإجراء عملية AND على محتويات المسجل reg مع محتويات مسجل التراكم A ونتيجة العملية توضع في المرمك . بنفس الطريقة يمكننا كتابة الصورة العامة للأمرين OR و XOR كما يلي :

ORA reg

XRA reg

حيث سيقوم الأمر الأول بإجراء عملية OR على محتويات المسجل reg مع محتويات المسجل A ويضع النتيجة في المسجل A . وأما الأمر الثاني فسيجري عملية XOR على محتويات المسجل reg مع محتويات المرمك ويضع النتيجة في المرمك . جميع العمليات المنطقية الثلاث السابقة يمكن أن تجرى على معلومة فورية أو ثابت وفي هذه الحالة فإن الصورة العامة لهذه الأوامر ستكون :

ANI data8

ORI data8

XRI data8

حيث data8 هو الثابت أو المعلومة الفورية المراد إجراء العملية المنطقية عليها. لاحظ أن هذه الأوامر في هذه الحالة سيكون كل منها مكونا من 2 بايت ، واحدة هي شفرة الأمر والثانية هي قيمة الثابت data8 .

مثال 4-17

افترض أن محتويات المسجل A تساوى FFH ومحتويات المسجل B تساوى FOH ، المطلوب إجراء عمليات AND و OR و XOR على كل من المسجلين A و B .

أولا : عملية AND

A=11111111

B=11110000

النتيجة بعد إجراء الأمر ANA B هي : 11110000

أى أن محتويات المرمك ستتغير إلى FOH . جميع الأعلام ستتأثر بهذه النتيجة ما عدا علمى الحمل والحمل النصفى حيث يكونان صفرا دائما فى جميع العمليات المنطقية لأنه لا يحدث لا حمل ولا حمل نصفى مع أى عملية منطقية .

ثانيا : عملية OR

A=11111111

B=11110000

النتيجة بعد إجراء الأمر OR A B هي : 11111111

أى أن محتويات المرجم ستظل FFH وستتأثر الأعلام بنفس الطريقة التى ذكرناها مع عملية AND .
ثالثا : عملية XOR

$$A=11111111$$

$$B=11110000$$

النتيجة بعد إجراء الأمر XRA B هى : 00001111

أى أن محتويات المرجم ستصبح 0FH وستتأثر جميع الأعلام بنفس الطريقة السابقة .

بذلك نكون قد أنهينا دراسة المجموعات الأساسية فى أوامر المعالج 8085 على أننا لم نذكر بالطبع جميع الأوامر داخل كل مجموعة ولكننا بذلك نكون قد عرفنا الغالبية العظمى من الأوامر وما تبقى منها فمن السهل معرفته من جداول الأوامر الموجودة فى آخر هذا الفصل كما أن هناك أبواب خاصة سستعرض لمجموعات معينة من الأوامر مثل الأوامر الخاصة بالبرامج الفرعية والأوامر الخاصة بالمقاطعة حيث سيفرد فصل خاص لكل منها إن شاء الله لأهميتها .

10-4 كيفية الإتصال بالذاكرة

Memory addressing

أى معلومة من حيث المكان إما أن تكون موجودة داخل المعالج نفسة فى أى مسجل من مسجلاته وفى هذه الحالة فإنه يمكن التعامل معها بسهولة وتحريكها من مكان لآخر داخل المعالج فى زمن أقل ، وإما أن تكون موجودة فى الذاكرة ويريد المعالج قراءتها ، أى إحضارها من الذاكرة ووضعها فى أى مسجل من مسجلاته ، أو كتابتها أى نقلها إلى الذاكرة ، وذلك يتطلب من المعالج أن يحدد عنوان المكان أو البايث فى الذاكرة التى ستتم معها عملية القراءة أو الكتابة . طريقة تحديد عنوان البايث من الذاكرة المراد التعامل معها هى المقصود دراسته فى هذا الجزء . يجب أن نتذكر جيدا أن جميع شرائح المعالجات التى نتعامل معها إلى الآن لها مسار عناوين به 16 بتا ولذلك فإننا عندما سنكتب أى عنوان فى النظام الستعشرى فلا بد أن نكتبه من 4 خانات حيث كل خانة فى النظام الستعشرى تمثل 4 بتات فى النظام الثنائى ، فمثلا العنوان التالى 1111000010100001 فى النظام الثنائى هو FOA1H فى النظام الستعشرى وغالبا نضع الحرف H للدلالة على أن الرقم مكتوبا فى النظام الستعشرى . هناك طريقتان يحدد بهما المعالج 8085 عنوان المكان أو البايث فى الذاكرة المراد التعامل معه وسنبين هاتين الطريقتين فيما يلى :

4-10-1 الطريقة المباشرة Direct method

في هذه الطريقة فإن الأمر نفسه يحتوى عنوان الباييت المراد التعامل معها ، ولذلك فإن كل الأوامر التى تقع تحت هذا الصنف تتكون من ثلاث بايتات ، واحدة من هذه البايئات هي شفرة الأمر operation code واختصارا تكتب op code والإثنين بايت التالية هي عنوان المكان في الذاكرة المراد التعامل معه . تذكر أن الباييت تتكون من 8 بتات وأنه دائما تكون الباييت الأولى من بايتات العنوان هي الباييت ذات القيمة الصغرى Low significant byte . بالنسبة للمعالج 8085 هناك الأوامر LDA addr و STA addr كأمثلة على الأوامر التى تتعامل بهذه الطريقة . كما رأينا سابقا فإن الأمر LDA addr يقوم بتحميل المسجل A (المركم) بمحتويات المكان الذى عنوانه موجود فى الأمر نفسه وقد رمزنا له بالرمز addr . أما الأمر STA addr فإنه يفعل العكس حيث يقوم بتخزين محتويات المسجل A فى المكان الذى عنوانه addr فى الذاكرة .

4-10-2 الطريقة غير المباشرة Indirect method

في هذه الطريقة يوضع عنوان الباييت المراد التعامل معها فى المسجلين H و L بحيث يحتوى المسجل H على الباييت ذات القيمة العظمى من العنوان ويحتوى المسجل L على الباييت ذات القيمة الصغرى منه . مثل هذه الأوامر تتكون دائما من بايت واحدة حيث يرمز لهذا المكان فى الذاكرة بالرمز M وتأخذ الشفرة 110 كما رأينا من خلال تعاملنا مع الأوامر سابقا . فمثلا الأمر MOV A,M معناه انقل محتويات باييت الذاكرة التى عنوانها فى زوج المسجلات HL إلى مسجل التراكم .

السؤال الذى يتبادر إلى الذهن هنا أى الطريقتين يفضل فى الاستخدام ، الطريقة المباشرة أم غير المباشرة ؟ الإجابة عن هذا السؤال تتوقف على البرنامج أو على المشكلة التى نبرمجها . فإذا كان البرنامج يتعامل مع الذاكرة باستمرار وبالذات إذا كان التعامل مع أماكن متعاقبة فى الذاكرة فإن الطريقة غير المباشرة لا شك تكون الأفضل لأنها ستوفر الكثير من طول البرنامج لأن المسجلين HL فى هذه الحالة سيكونان عبارة عن مؤشر إلى باييت الذاكرة التى تستخدمها وكلما أردت التعامل مع بايت جديدة تزيد محتويات المسجلين HL بواحد كما رأينا فى المثال 4-7 . أما إذا كنت ستتعامل مع الذاكرة لمرة واحدة فإنه ليس هناك أى داع لاستخدام الطريقة غير المباشرة ولكن يفضل فى هذه الحالة الطريقة المباشرة .

الأشكال التالية تحتوى مجموعة أوامر المعالج 8085 مقسمة أولا إلى مجموعات كما أشرنا سابقا بحيث يحتوى كل شكل الأوامر الخاصة بمجموعة معينة

والشفرة الست عشرية وعدد نبضات التزامن التي يحتاجها كل أمر لكي يتم إحضاره من الذاكرة وتنفيذه . شكل (4-18) يبين قائمة أوامر الشريحة 8085 مرتبة ترتيبا أبجديا مع ملخص أو نبذة عن وظيفة كل أمر وكذلك الأعلام التي تتأثر بكل أمر .

MOV	A	B	C	D	E	H	L	M
,A	7F	47	4F	57	5F	67	6F	77
,B	78	40	48	50	58	60	68	70
,C	79	41	49	51	59	61	69	71
,D	7A	42	4A	52	5A	62	6A	72
,E	7B	43	4B	53	5B	63	6B	73
,H	7C	44	4C	54	5C	64	6C	74
,L	7D	45	4D	55	5D	65	6D	75
,M	7E	46	4E	56	5E	66	6E	

MVI	A	B	C	D	E	H	L	M
	3Exx	06xx	0Exx	16xx	1Exx	26xx	2Exx	36xx

LDA (13)	STA (13)
3Axxxx	32xxxx

LDAX B (7)	LDAX D (7)	STAX B (7)	STAX D (7)
0A	1A	02	12

IN (10)	OUT (10)	LHLD (16)	SHLD (16)
DBxx	D3xx	2Axxxx	22xxxx

XCHG (4)	XTHL (16)	SPHL (6)
EB	E3	F9

	LXI (10)	POP (10)	PUSH (10)
B	01xxxx	C1	C5
D	11xxxx	D1	D5
H	21xxxx	E1	E5
SP	31xxxx		
PSW		F1	F5

- جميع الأوامر التي على الصورة MOV M,reg و MOV reg,M تأخذ 7 نبضات تزامن والأوامر التي على الصورة MOV reg,reg تأخذ 4 نبضات تزامن .
- xx تعني معلومة 8 بتات (data8) و xxxx تعني عنوان أو معلومة 16 بتا
- جميع الأوامر التي على الصورة MVI reg,xx تأخذ 7 نبضات تزامن حتى يتم إحضارها وتنفيذها والأمر MVI M,xx يأخذ 10 نبضات.
- الأرقام التي بين القوسين للأوامر الأخرى تدل على عدد النبضات التي يأخذها الأمر.

شكل (4-13) مجموعة أوامر الانتقال للمعالج 8085

	IN R	DCR	ADD	SUB	ADC	SBB
A	3C	3D	87	97	8F	9F
B	04	05	80	90	88	98
C	0C	0D	81	91	89	99
D	14	15	82	92	8A	9A
E	1C	1D	83	93	8B	9B
H	24	25	84	94	8C	9C
L	2C	2D	85	95	8D	9D
M	34	35	86	96	8E	9E

	INX	DCX	DAD
B	03	0B	09
D	13	1B	19
H	23	2B	29
SP	33	3B	39

ADI(7)	SUI(7)	ACI(7)	SBI(7)	DAA(4)
C6xx	D6xx	CExx	DExx	27

- الأوامر ADD M , SUB M , ADC M , SBB M كلها تأخذ 7 نبضات لكي يتم إحضارها وتنفيذها .
- الأمران INR M DCR M تأخذ 10 نبضات .
- جميع أوامر الحساب الأخرى تحتاج إلى 4 نبضات .
- الأرقام التي بين القوسين لبعض الأوامر تدل على عدد النبضات اللازمة لإحضار وتنفيذ هذا الأمر .

شكل (4-14) مجموعة أوامر الحساب

	ANA	ORA	XRA	CMP
A	A7	B7	AF	BF
B	B0	A0	A8	B8
C	A1	B1	A9	B9
D	A2	B2	AA	BA
E	A3	B3	AB	BB
H	A4	B4	AC	BC
L	A5	B5	AD	BD
M	A6	B6	AE	BE

ANI	ORI	XRI	CPI	CMA
E6xx	F6xx	EExx	FExx	2F

- جميع أوامر المنطق تأخذ 4 نبضات ما عدا الأوامر التي تتعامل مع ذاكرة M والأوامر ANI ORI XRI CPI فتأخذ 7 نبضات لكي يتم إحضارها وتنفيذها.

شكل (4-15) مجموعة أوامر المنطق

JMP	JC	JNC	JZ	JNZ	JM	JP	JPE	JPO
C3 XXXX	DA XXXX	D2 XXXX	CA XXXX	C2 XXXX	FA XXXX	F2 XXXX	EA XXXX	E2 XXXX

• هذه الأوامر إذا تم لها القفز تأخذ 10 نبضات وإذا لم يتم القفز تأخذ 7 نبضات لكي يتم إحضارها وتنفيذها

CALL	CC	CNC	CZ	CNZ	CM	CP	CPE	CPO
CD XXXX	DC XXXX	D4 XXXX	CC XXXX	C4 XXXX	FC XXXX	F4 XXXX	EC XXXX	E4 XXXX

• هذه الأوامر تأخذ 18 نبضة إذا تم لها القفز وإذا لم يتم القفز تأخذ 9 نبضات .

RET	RC	RNC	RZ	RNZ	RM	RP	RPE	RPO
C9	D8	D0	C8	C0	F8	F0	E8	E0

• هذه الأوامر تأخذ 12 نبضة إذا تم القفز و 6 إذا لم يتم ما عدا الأمر RET فإنه يأخذ 10 نبضات .

RST 7	RST 6	RST 5	RST 4	RST 3	RST 2	RST 1	RST 0
FF	F7	EF	E7	DF	D7	CF	C7

• هذه الأوامر تأخذ 12 نبضة لكي يتم إحضارها وتنفيذها .

PCHL
E9

• هذا الأمر يحتاج إلى 6 نبضات لكي يتم إحضاره وتنفيذه .

شكل (4-16) مجموعة أوامر القفز

RLC	RRC	RAL	RAR	CMC	STC	EI	DI	HLT	NOP
07	0F	17	1F	3F	37	FB	F3	76	00

RIM	SIM
20	30

• جميع هذه الأوامر تحتاج إلى 4 نبضات لكي يتم إحضارها وتنفيذها .

شكل (4-17) أوامر أخرى

الاسمبلى	الشفرة	الأعلام المتأثرة	وظيفة الأمر
ACI const	11001110 data8	Z S P C Y H C	ثابت+علم الحمل+ A ← A
ADD reg	10000sss	Z S P C Y H C	مسجل ← A + A
ADC reg	10001sss	Z S P C Y H C	مسجل+علم الحمل+ A ← A
ADI const	11000110 data8	Z S P C Y H C	ثابت ← A + A
ANA reg	10100sss	Z S P 0 0	مسجل ← A AND A
ANI const	11100110 data8	Z S P 0 0	ثابت ← A AND A
CALL addr	CD addr16		نداء غير مشروط على برنامج فرعى
CC addr	DC addr16		نداء برنامج فرعى مشروط بعلم الحمل=1
CM addr	FC addr16		نداء برنامج فرعى مشروط بنتيجة سالبة
CMA	2F		اعكس محتويات المسجل A
CMC	3F		اعكس علم الحمل
CMP reg	10111sss	Z S P C Y H C	قارن مسجل مع مسجل A. اطرح (A - reg) المسجل A لا يتأثر بالنتيجة.
CNC addr	D4 addr16		نداء برنامج فرعى مشروط بعلم الحمل=0
CNZ	C4 addr16		نداء برنامج فرعى مشروط بعلم الصفر=0
CP addr	F4 addr16		نداء برنامج فرعى مشروط بنتيجة موجبة
CPE addr	EC addr16		نداء برنامج فرعى مشروط بباريتى زوجية
CPI const	FE data8	Z S P C Y H C	مقارنة , (ثابت - A) والمسجل A لا يتأثر
CPO addr	E4 addr16		نداء برنامج فرعى مشروط بباريتى فردية
CZ addr	CC addr16		نداء برنامج فرعى مشروط بعلم الصفر=1
DAA	27	Z S P C Y H C	حول المركم إلى النظام العشري
DAD rp	00rp1001	CY	اجمع HL ← HL + rp
DCR reg	00ddd101	Z S P C Y H C	انقص محتويات المسجل reg بمقدار 1
DCX rp	00rp1011		انقص محتويات المسجلين rp بمقدار 1
DI	F3		اهمل المقاطعة
EI	FB		اسمح بالمقاطعة

HLT	76		أوقف تنفيذ البرنامج
IN no.	DB Port no.		اقرأ بوابة الإدخال رقم no.
INR reg	00ddd100	Z S P CY HC	اجمع 1 على محتويات المسجل reg
INX rp	00rp0011		اجمع 1 على محتويات المسجلين rp
JC addr	DA addr16		قفز مشروط بعلم الحمل = 1
JM addr	FA addr16		قفز مشروط بعلم الإشارة = 1
JMP addr	C3 addr16		قفز غير مشروط
JNC addr	D2 addr16		قفز مشروط بعلم الحمل = 0
JNZ addr	C2 addr16		قفز مشروط بعلم الصفر = 0
JP addr	F2 addr16		قفز مشروط بعلم الإشارة = 0
JPE addr	EA addr16		قفز مشروط بباريتي زوجية
JPO addr	E2 addr16		قفز مشروط بباريتي فردية
JZ addr	CA addr16		قفز مشروط بعلم الصفر = 1
LDA addr	3A addr16		محتويات عنوان ← المسجل A
LDAX rp	00rp1010		محتويات العنوان الموجود في الزوج BC أو DE تنقل للمسجل A
LHLD addr	2A addr16		محتويات العنوان addr والذي يليه ← المسجلين HL
LXI rp,const	00rp0001 data16		ثابت من 16 بت ← زوج المسجلات rp
MOVreg1,reg2	01dddsss		محتويات reg2 ← reg1
MVIreg,const	00ddd110 data8		ثابت من 8 بت ← reg
NOP	00		لا تعمل شيء
ORA reg	10110sss	Z S P 0 0	OR المسجل reg مع المسجل A
ORI const	F6 data8	Z S P 0 0	OR ثابت من 8 بت مع المسجل A

OUT no.	D3 Port no.		المسجل A ← بوابة الإخراج رقم no.
PCHL	E9		قفز للعنوان الموجود في المسجلين HL
POP rp	11rp0001		محتويات قمة المكدة ← rp
PUSH rp	11rp0101		محتويات rp ← قمة المكدة
RAL	17	CY	دوران المركم للشمال من خلال علم الحمل
RAR	1F	CY	دوران المركم لليمين من خلال علم الحمل
RC	D8		عودة مشروطة بعلم الحمل = 1
RET	C9		عودة غير مشروطة
RLC	07	CY	دوران المركم للشمال ، آخر بت الى البت الأولى وإلى علم الحمل ولا يذهب علم الحمل لأول بت
RM	F8		عودة مشروطة بنتيجة سالبة
RNC	D0		عودة مشروطة بعلم الحمل = 0
RNZ	C0		عودة مشروطة بعلم الصفر = 0
RP	F0		عودة مشروطة بنتيجة موجبة
RPE	E8		عودة مشروطة ببازيتي زوجية
RRC	0F	CY	دوران المركم لليمين ، أول بت إلى آخر بت وإلى علم الحمل ولا يذهب علم الحمل لآخر بت
RST	11nnn111		ابدأ من العنوان صفر
RZ	C8		عودة مشروطة بعلم الصفر = 1
RIM	20		اقرأ قناع المقاطعة
SBB reg	10010sss	Z S P C Y H C	طرح reg وعلم الحمل من المسجل A
SBI const	DE data8	Z S P C Y H C	اطرح ثابت وعلم الحمل من المسجل A
SHLD	22		خزن محتويات المسجلين HL في الذاكرة
SIM	30		ضع قناع المقاطعة
SPHL	F9		حمل مؤشر المكدة من المسجلين HL
STA add	32 addr16		خزن محتويات المركم في العنوان addr
STAX rp	00rp0010		خزن المركم في العنوان الموجود في المسجلين rp (BC و DE فقط)
STC	37	--- 1 -	اجعل علم الحمل = 1
SUB reg	10010sss	Z S P C Y H C	اطرح المسجل reg من المسجل A
SUI const	D6 data8	Z S P C Y H C	اطرح ثابت من المسجل A
XCHG	EB		بدل محتويات المسجلين DE مع HL
XRA reg	10101sss	Z S P 0 0	XOR المسجل reg مع المركم

XRI const	EE data8	ZSP00	XOR ثابت مع المرمك
XCHG	EB		بدل محتويات المسجلين HL مع مؤشر المكسدة

ملاحظات :

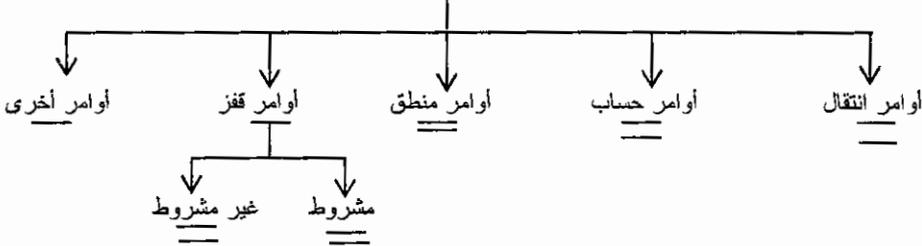
- **reg** اختصار لكلمة register وتعني مسجل 8 بت .
- **rp** اختصار لكلمة register pair وتعني زوجا من المسجلات .
- **const** اختصار لكلمة constant وتعني ثابت أو معلومة فورية وأحيانا يكون هذا الثابت 8 بتات وأحيانا يكون 16 بتا على حسب الأمر إذا كان يتعامل مع مسجل واحد أو زوج من المسجلات .
- **addr** اختصار لكلمة address وتعني عنوانا في الذاكرة ودائما يكون العنوان 16 بتا .
- **Z** تعني علم الصفر S
- تعني علم الإشارة
- **P** تعني علم الباريتي
- **CY** تعني علم الحمل **HC** تعني علم الحمل النصفى .

شكل (4. 18) أوامر الشريحة 8085 مرتبة أبجديا

11-4 تمارين

1. أكمل الجدول التالي الخاص بأوامر الشريحة 8085 :

أوامر المعالج 8085



2. ما هي نتيجة تنفيذ البرنامج التالي :

```

E000 MVI A,05
E002 MOV B,A
E003 MOV C,B
E004 MOV D,C
E005 MOV E,D
E006 MOV L,E
E007 MOV H,L
E008 MOV M,L
  
```

3. اقرأ البرنامج السابق وأجب عما يلي :

- محتويات مكان الذاكرة=E000
- محتويات مكان الذاكرة=E001
- محتويات مكان الذاكرة=0505

4.

E000 LXI H,E100
E003 MVI M,3E
E005 INX H
E006 MVI M,05
E008 INX H
E009 MVI M,47
E00B INX H
E00C MVI M,48

ما هي نتيجة تنفيذ البرنامج السابق ؟

5. على ضوء نتيجة تنفيذ البرنامج السابق ما هي نتيجة تنفيذ الشفرات الموجودة في الأماكن E100 إلى E103 ؟

6. هل تتأثر الأعلام بأوامر الانتقال ؟

7. اذكر الأعلام التي تتأثر بكل عملية من العمليات الحسابية والمنطقية ؟

8. إذا كانت محتويات المسجل A=F3H ومحتويات المسجل B=A4H فاكتب محتويات المسجل A بعد تنفيذ كل أمر من الأوامر التالية على نفس المحتويات السابقة ووضح أيضا كيف ستتأثر الأعلام بكل أمر :

ADD B
SUB B
SUB A
INC A
ANA B
ORA B
XRA B

9. ارسم خريطة تدفق للبرنامج التالي وما هي نتيجة تنفيذه :

E000 MVI L,50H
E003 MVI H,E1H
E005 MOV M,A
E006 DCR L
E007 JNZ E005

10. ماذا يحدث لو كتبنا البرنامج السابق عند E100 بدلا من E000 ؟

11. أعد كتابة البرنامج السابق مستخدما العلامات Labels ؟ وما هي مميزات البرنامج مكتوبا بهذه الصورة ؟

12. كم عدد بوابات الإدخال التي يستطيع المعالج 8085 التعامل معها؟
13. كم عدد بوابات الإخراج التي يستطيع المعالج 8085 التعامل معها؟
14. على ماذا يتوقف هذا العدد ؟
15. هل هناك ما يمنع أن تكون بوابتي إدخال وإخراج لهما نفس الرقم ، كمثال على ذلك IN 05 و OUT 05 ؟
16. OUT ©,reg هذا أحد أوامر الإخراج للبروسيسور Z80 والذي يعنى إخراج محتويات المسجل reg على بوابة الإخراج التي رقمها فى المسجل C فهل المعالج 8085 لديه ما يكافىء هذا الأمر ؟
17. هل تتأثر الأعلام بأوامر الإدخال والإخراج ؟
18. أكتب برنامجاً يقرأ محتويات البوابة 00 وإذا كانت هذه المحتويات زوجية يخزنها فى الذاكرة ابتداءً من العنوان E100 وإذا كانت فردية يخرجها على البوابة 00 ؟
19. المعالج 8085 لديه طريقتان فقط لعنونة الذاكرة وهما العنونة المباشرة والعنونة غير المباشرة ، اذكر الأوامر التي تستخدم مع كل من الطريقتين ؟
20. اذكر متى تفضل استخدام العنونة المباشرة ومتى تفضل العنونة غير المباشرة؟
21. المعالجان Z80 و MC6800 بهما طرق أخرى لعنونة الذاكرة والتي منها على سبيل المثال العنونة المفهرسة indexed addressing فهل هناك طرق أخرى لعنونة الذاكرة لدى المعالج 8085 ؟
22. اكتب برنامجاً يحسب عدد الواحد الموجودة فى محتويات المسجل A ، فمثلاً إذا كان $A=11110101$ فإن عدد الواحد = 6 .
23. اكتب برنامجاً يحسب أكبر قيمة عددية فى أى بايت فى المدى العنوانى E200H إلى E250H .
24. اكتب برنامجاً يحسب عدد البايتات التي تحتوى أصفراً والتي تحتوى أرقاماً موجبة والتي تحتوى أرقاماً سالبة فى المدى العنوانى E100 إلى E150 .
25. اكتب برنامجاً يحسب عدد البايتات التي تحتوى بيانات فردية والتي تحتوى بيانات زوجية فى المدى العنوانى E100 إلى E150 .
26. المدى العنوانى E100 إلى E150 يحتوى بيانات لإشارة صوت ، احسب كم مرة عبرت إشارة الصوت الصفرة .
27. اكتب برنامجاً يقرأ بوابة الإدخال رقم 00 ويختبر البت الرابعة فيها ، فإذا كانت هذه البت واحد يخرج هذه المحتويات على البوابة 00 ، وإذا كانت هذه البت صفر يخرج محتوياتها على البوابة 01 .