بنيان الحواسب

جلسة عملي

إعداد: م.همام ياسين

إشراف: د.فادي متوج

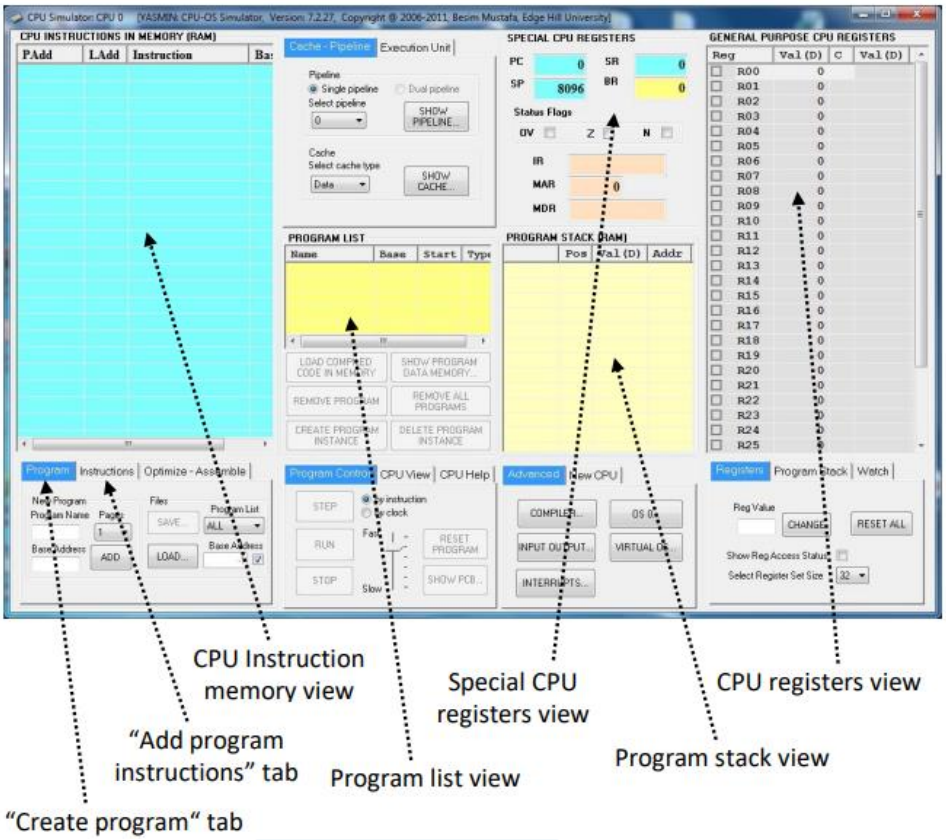
1. الهدف من المحاضرة:

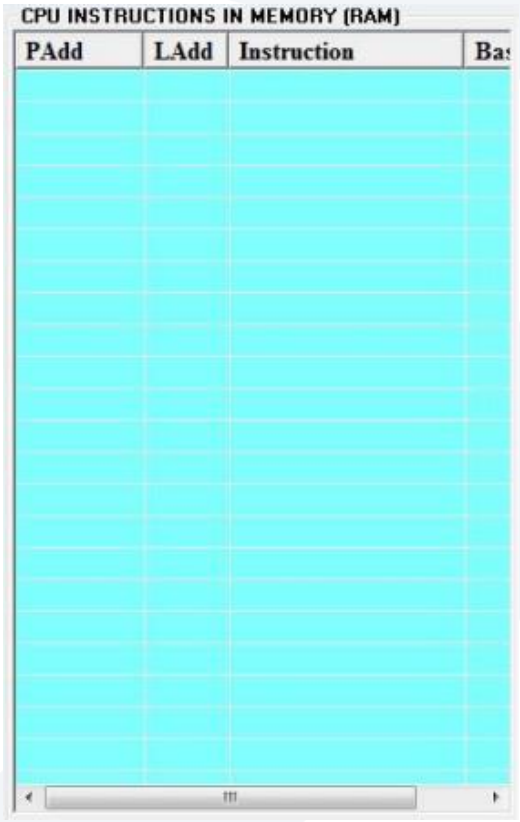
في نهاية المحاضرة يجب أن تكون قادراً على:

- استخدام المحاكي CPU Simulator لإنشاء التعليمات الأساسية الخاصة بالمعالج.
- استخدام المحاكي لتنفيذ تلك التعليمات.
- التعامل مع تعليمات نقل البيانات إلى المسجلات، ومقارنة القيم في المسجلات، والانتقال إلى مواقع العناوين وإضافة قيم إلى القيم الموجودة في المسجلات.
- شرح وظيفة المسجلات الخاصة لوحدة المعالجة المركزية، وهي PC (عداد البرامج) و SR (مسجل الحالة) وأعلام الحالة OV و N و Z.
- تنفيذ برنامج يحوي عبارات شرطية بسيطة.

2. تفاصيل المحاكي

تتكون النافذة الرئيسية من أقسام تمثل أجزاء وظيفية مختلفة لوحدة المعالجة المركزية التي تمت محاكاتها. هذه الأجزاء موضحة في الشكل التالي.





2.1. قسم ذاكرة تعليمات وحدة المعالجة المركزية: (CPU instruction memory view)

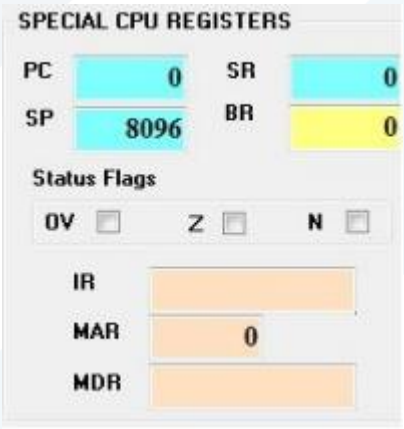
يحتوي هذا القسم على تعليمات البرنامج. يتم عرض التعليمات كسلسلة من التعليمات منخفضة المستوى (بصيغة لغة التجميع Assembly) وليست كرموز ثنائية، وذلك من أجل التوضيح وجعل الكود أكثر قابلية للقراءة من قبل البشر.

ترتبط كل تعليمات بعنوانين: العنوان الفعلي (PAdd) والعنوان المنطقي (LAdd).

كما يعرض هذا القسم العنوان الأساسي (Base) مقابل كل تعليمة،

ملاحظة: سيكون لسلسلة التعليمات التي تنتمي إلى نفس البرنامج نفس العنوان الأساسي.

2.2. قسم مسجلات وحدة المعالجة المركزية الخاصة: Special CPU registers View



يُظهر هذا القسم مجموعة مسجلات وحدة المعالجة المركزية ذات الوظائف المخصصة والمحددة مسبقاً:

PC عداد البرنامج: يحتوي على عنوان التعليمة التالية التي سيتم تنفيذها.

IR مسجل التعليمات: يحتوي على التعليمات التي يتم تنفيذها حاليًا.

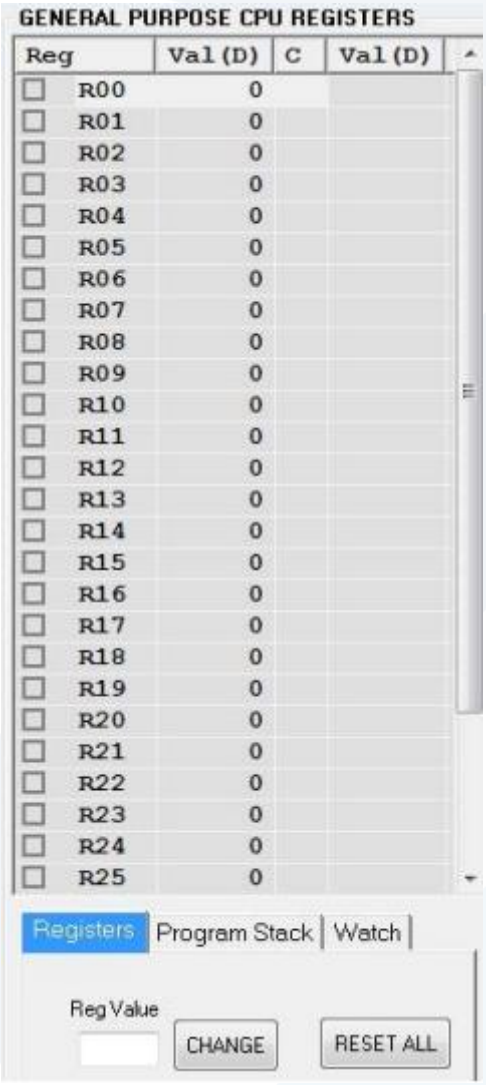
SR مسجل الحالة: يحتوي على معلومات تتعلق بنتيجة آخر تعليمة تم تنفيذها.

SP مؤشر المكدّس: مسجل يشير إلى القيمة المحفوظة في أعلى مكدّس البرنامج.

BR المسجل الأساسي: يحتوي على العنوان الأساسي الحالي.

MAR مسجل عنوان الذاكرة: يحتوي على عنوان الذاكرة الذي يتم الوصول إليه حاليًا.

بتات الحالة: OV: تجاوز (طفحان) ؛ Z: صفر ؛ N: سلبي.



2.3. قسم مسجلات وحدة المعالجة المركزية: CPU registers View

يُظهر هذا القسم محتويات مسجلات الأغراض العامة، والتي تُستخدم لتخزين القيم المؤقتة عند تنفيذ تعليمات البرنامج.

المسجلات هي ذواكر سريعة جدًا تحمل قيمًا مؤقتة أثناء تنفيذ وحدة المعالجة المركزية للتعليمات.

تدعم هذه البنية من 8 إلى 64 مسجلاً. غالبًا ما تستخدم هذه المسجلات للاحتفاظ بقيم متغيرات البرنامج كما هو محدد في اللغات عالية المستوى.

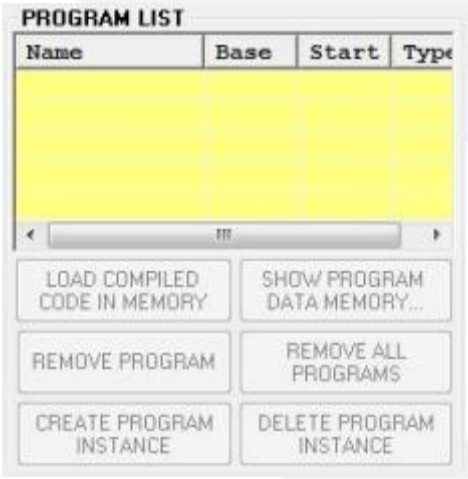
يختلف عدد هذه المسجلات بحسب بنية المعالج المستخدم، فبعض المعالجات قد يملك 8 مسجلات فقط، وبعضها الآخر قد يملك 128 مسجلاً للاغراض العامة، لكن في جميع الحالات فإن هذه المسجلات لها نفس الوظيفة.

يعرض في هذا القسم اسم كل سجل (Reg) وقيمته الحالية (Val) وبعض القيم الإضافية المحجوزة لتصحيح أخطاء البرنامج. يمكن استخدامه أيضًا لإعادة ضبط السجل الفردي يدويًا والتي غالبًا ما تكون مفيدة لتصحيح الأخطاء المتقدم.

2.4. قسم مكدّس البرنامج: Program stack View

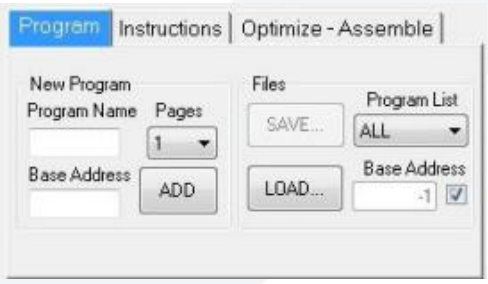
مكدس البرنامج هو منطقة أخرى تحافظ على القيم المؤقتة أثناء تنفيذ التعليمات. المكدس عبارة عن بنية بيانات LIFO (الداخل أخيراً يخرج أولاً). كما أنها تستخدم لإجراء المقاطعة واستدعاء التوابع الفرعية بكفاءة. كل برنامج له مكدس فردي خاص به.

تُستخدم تعليمات وحدة المعالجة المركزية PSH (الدفع) و POP لتخزين القيم أعلى قيم المكدس وسحب القيم من أعلى المكدس على التوالي.



2.5. قسم قائمة البرنامج: Program list View .

يستخدم زر إزالة البرنامج لإزالة البرنامج المحدد من القائمة ؛ استخدم الزر REMOVE ALL PROGRAMS لإزالة كافة البرامج من القائمة. ويمكن عرض ذاكرة معطيات برنامج محدد بواسطة الزر SHOW PROGRAM DATA MEMORY.



2.6. إنشاء البرنامج: Program creation

لإنشاء برنامج جديد نكتب اسمه في مربع اسم البرنامج وعنوانه الأساسي في مربع العنوان الأساسي ثم ننقر فوق الزر ADD. سيظهر اسم البرنامج الجديد في قسم قائمة البرامج المذكور في الفقرة السابقة.

2.7. التعامل مع التعليمات: Instructions

زر ADD NEW لإضافة تعليمة جديدة.

زر EDIT لتعديل التعليمة المختارة.

أزرار MOVE DOWN / MOVE UP لتحريك التعليمة المحددة لأسفل أو لأعلى.

أزرار INSERT BELOW / INSERT ABOVE... لإدخال تعليمة جديدة أعلى أو أسفل التعليمة المحددة على التوالي.

3. تمارين

لكتابة أي برنامج نبدأ من قسم Program في الفقرة 6 السابقة، نكتب اسماً للبرنامج في Program Name (وليكن الاسم First)، ثم نضيف عنواناً أساسياً Base Address (يمكن أن نكتب أي رقم وليكن 100)، ثم نضغط على زر ADD. بهذا يضاف البرنامج First في قائمة البرامج Program List. (يمكن حفظ البرنامج في ملف Save، أو تحميل برنامج محفوظ سابقاً من ملف Load).

بعد إنشاء البرنامج أصبحنا جاهزين لإدخال التعليمات في القسم Instructions في الفقرة 7 السابقة، وذلك بالضغط على زر ADD NEW مما يفتح نافذة جديدة Instructions: CPU 0 (استخدم ملحق التعليمات 4 في نهاية المحاضرة).

تمرين 1:

1. أنشئ تعليمة تنقل الرقم 5 إلى المسجل R00، ثم نفّذها بالنقر عليها مرتين وراقب مسجلات الأغراض العامة.
2. أنشئ تعليمة تنقل الرقم 7 إلى المسجل R01، ثم نفّذ هذه التعليمة وراقب مسجلات الأغراض العامة أيضاً.
3. أنشئ تعليمة تضيف محتوى المسجلين R00 و R01، ثم نفّذها، وراقب في اي مسجل تم وضع النتيجة.
4. أنشئ تعليمة تقفز إلى التعليمة الأولى، ثم نفّذها، وراقب القيمة في عداد البرنامج PC (عنوان التعليمة التالية التي ستنفّذ).
5. أدخل يدوياً قيمتين متساويتين في كل من المسجلين R04 و R05، ثم أنشئ تعليمة لمقارنة القيم في المسجلين ونفّذها.
6. راقب قيم أعلام الحالة OV,Z,N، ماذا تغير فيها؟
7. أدخل يدوياً قيمة في المسجل R04 أكبر من القيمة في المسجل R05، ثم أعد تنفيذ التعليمة السابقة.
8. راقب قيم أعلام الحالة OV,Z,N، ماذا تغير فيها؟
9. أدخل يدوياً قيمة في المسجل R04 أقل من القيمة في المسجل R05، ثم أعد تنفيذ التعليمة السابقة.
10. راقب قيم أعلام الحالة OV,Z,N، ماذا تغير فيها؟
11. أنشئ تعليمة تقفز إلى التعليمة الأولى إذا تساوت القيم في كل من المسجلين R04 و R05.
12. اختبر التعليمة السابقة بإدخال قيمتين متساويتين في كل من المسجلين، ثم إعادة تنفيذ تعليمة المقارنة، ثم تنفيذ تعليمة القفز (إذا تمت عملية المقارنة بنجاح سيحدد البرنامج التعليمة الأولى).

CPU INSTRUCTIONS IN MEMORY (RAM)

PAdd	LAdd	Instruction
0000	0000	MOV #5, R00
0006	0006	MOV #7, R01
0012	0012	ADD R00, R01
0017	0017	JMP 0
0021	0021	CMP R04, R05
0026	0026	JZR 0

تمرين 2: الحلقات:

1. أنشئ تعليمة تنقل الرقم 0 إلى المسجل R01.
2. أنشئ تعليمة تضيف الرقم 1 إلى المسجل R01.
3. أنشئ تعليمة تقارن الرقم 10 مع المسجل R01.
4. أنشئ تعليمة تقفز إلى التعليمة 2 إذا لم تتساوى قيمة كل من R01 و الرقم 10.
5. أنشئ تعليمة HLT.
6. نفّذ التعليمات من 1 حتى 4 بالتتالي. ماذا تلاحظ؟
7. اضغط على RESET PROGRAM، ستحدد التعليمة رقم 1، ثم اضغط RUN. لاحظ سير الحلقة وما النتيجة؟

CPU INSTRUCTIONS IN MEMORY (RAM)

PAdd	LAdd	Instruction
0000	0000	MOV #0, R01
0006	0006	ADD #1, R01
0012	0012	CMP #10, R01
0018	0018	JNE 6
0022	0022	HLT

تمرين 3:

أنشئ كود برمجي لعبارة شرطية، إذا كانت القيمة في المسجل R02 أكبر من القيمة في المسجل R01، عندها ضع القيمة 8 في المسجل R03. نفذ البرنامج.

CPU INSTRUCTIONS IN MEMORY (RAM)

PAdd	LAdd	Instruction
0100	0000	CMP R01, R02
0105	0005	JGT 13
0109	0009	JLE 19
0113	0013	MOV #8, R03
0119	0019	HLT

تمرين 4:

أنشئ كود برمجي لحلقة تتكرر 5 مرات، بحيث تزداد القيمة في المسجل R02 بمقدار 2 في كل تكرار. نفذ البرنامج.

CPU INSTRUCTIONS IN MEMORY (RAM)

PAdd	LAdd	Instruction
0100	0000	MOV #5, R03
0106	0006	CMP R03, R04
0111	0011	JGE 31
0115	0015	ADD #2, R02
0121	0021	ADD #1, R04
0127	0027	JMP 6
0131	0031	HLT

4. ملحق التعليمات:

Instruction	Description
Data transfer instructions	
MOV	Move a number to register; move register to register MOV #2, R01 moves number 2 into register R01 MOV R01, R03 moves contents of register R01 into register R03
LDB	Load a byte from memory to register
LDW	Load a word (2 bytes) from memory to register
STB	Store a byte from register to memory
STW	Store a word (2 bytes) from register to memory
PSH	Push a number to top of program stack (TOS); push register to TOS PSH #6 pushes number 6 on top of the stack PSH R03 pushes the contents of register R03 on top of the stack
POP	Pop data from top of program stack to register POP R05 pops contents of top of stack into register R05 Note: If you try to POP from an empty stack you will get the error message "Stack underflow"
Arithmetic instructions	
ADD	Add a number to register; add register to register ADD #3, R02 adds number 3 to contents of register R02 and stores the result in register R02. ADD R00, R01 adds contents of register R00 to contents of register R01 and stores the result in register R01.
SUB	Subtract number from register; subtract register from register
MUL	Multiply number with register; multiply register with register
DIV	Divide number with register; divide register with register

Control transfer instructions	
JMP	Jump to given instruction address unconditionally JMP 100 unconditionally jumps to instruction address location 100
JLT	Jump to instruction address if less than (after last comparison)
JGT	Jump to instruction address if greater than (after last comparison)
JEQ	Jump to instruction address if equal (after last comparison instruction) JEQ 200 jumps to instruction address location 200 if the previous comparison instruction result indicates that the two numbers are equal, i.e. the Z status flag is set (the Z box will be checked in this case).
JNE	Jump to instruction address if not equal (after last comparison)
CAL	Jump to subroutine address
RET	Return from subroutine
SWI	Software interrupt (used to request OS help)
HLT	Halt Simulation
Comparison instruction	
CMP	Compare a number with register; compare register with register CMP #5, R02 compare number 5 with the contents of register R02 CMP R01, R03 compare the contents of registers R01 and R03 Note: If R01 = R03 then the status flag Z will be set, i.e. the Z box is checked. If R01 < R03 then none of the status flags will be set, i.e. none of the status flag boxes are checked. If R01 > R03 then the status flag N will be set, i.e. the N status box is checked.
Input, output instructions	
IN	Get input data (if available) from an external IO device
OUT	Output data to an external IO device

انتهت المحاضرة