بنيان الحواسب

المحاضرة 5 عملي

إعداد: م.همام ياسين

إشراف: د.فادي متوج

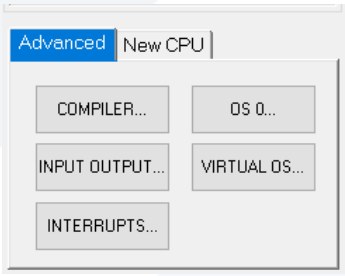
1. الهدف من المحاضرة:

في نهاية هذه المحاضرة يجب أن تكون قادرًا على:

- كتابة برنامج بلغة عالية المستوى ضمن المترجم (Compiler) وترجمته وتحميله في الذاكرة.
- فهم مبدأ أنبوبة المعالجة Pipelining مع تطبيق مثال عملي.
- معرفة الفرق بين التنفيذ المتسلسل والمتداخل للتعليمات.

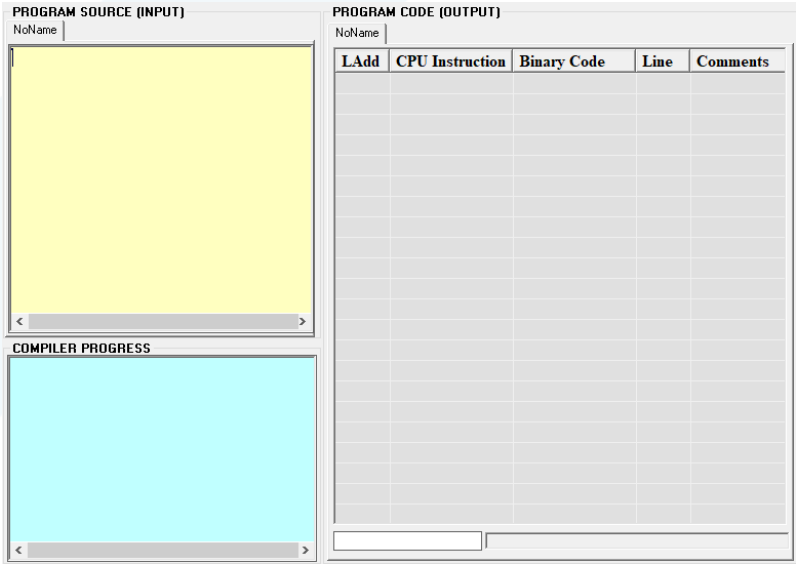
2. تفاصيل المحاكي

يمكن الوصول إلى المترجم من الخيار Advanced > Compiler.



الشكل 1 – نافذة فتح المترجم Compiler

تتكون نافذة المحاكي من عدة نوافذ فرعية هي نافذة البرنامج (باللون الأصفر) ويكتب ضمنها البرنامج باللغة عالية المستوى، ونافذة عملية الترجمة (باللون الأزرق) ويكتب فيها خطوات الترجمة التي يقوم بها برنامج CPU OS Simulator، ونافذة الكود (بالرمادي) وتحوي البرامج بلغى التجميع بعد ترجمته.



الشكل 2 – النوافذ الفرعية داخل خيار Copmiler

3. النظرية الأساسية

3.1. كتابة البرنامج بلغة عالية المستوى وتحويله إلى لغة التجميع:

تتألف عملية الترجمة من أربعة خطوات:

1- تحويل البيانات البرنامج المصدر إلى رموز (Lexical Analysis) Tokenizing source statements.

2- بناء جدول الرموز Constructing the symbol table.

3- التحقق من البرنامج المصدر قواعدياً (Syntax Analysis) Checking source syntax.

4- توليد الكود بلغة التجميع Generating code.

3.2. تقنية أنبوبة المعالجة Pipelining:

تتضمن وحدات المعالجة المركزية الحديثة خطوط تعليمات قادرة على معالجة مراحل مختلفة من التعليمات متعددة المراحل بالتوازي وبالتالي تحسين الأداء العام لوحدات المعالجة المركزية.

.لكن معظم البرامج تتضمن تعليمات لا تصلح بسهولة لتخضع لعملية أنبوبة المعالجة (التقسيم) مما يتسبب بمخاطرة فقد البيانات، ويقلل بشكل فعال من أداء وحدة المعالجة المركزية. نتيجة لذلك يتم تصميم خطوط أنابيب وحدة المعالجة المركزية بشكل مناسب لتصبح فادرة على التعامل مع هذه المخاطر.

أي يمكن القول أن عملية أنبوبة المعالجة Pipelining في المعالجات هي تقنية تستخدم لزيادة كفاءة تنفيذ التعليمات في وحدة المعالجة المركزية (CPU)، وهي تقنية شائعة جداً في تصميم المعالجات الحديثة.

وتتمثل فكرة أنبوبة المعالجة في تقسيم عملية تنفيذ التعليمات في المعالجة الحاسوبية إلى مراحل متعددة، وتشغيل كل مرحلة بشكل مستقل ومتزامن مع مراحل أخرى في نفس الوقت. وبهذه الطريقة، يمكن تنفيذ عدة تعليمات في نفس الوقت، وبالتالي زيادة سرعة تنفيذ البرامج.

يجدر بالذكر أن الخطوات الخمس للتعامل مع كل تعليمة في المعالجات هي:

1. إحضار التعليمة (Instruction Fetch): يقوم المعالج بقراءة التعليمة من الذاكرة (Memory) وتخزينها في السجل الداخلي للمعالج.
2. فك تشفير التعليمة (Instruction Decode): يقوم المعالج بتفسير التعليمة وفهم ماهي العملية التي يجب تنفيذها وماهي المعاملات المطلوبة لتنفيذها.
3. إحضار البيانات (Data Fetch): إذاكانت التعليمة تتطلب بيانات من الذاكرة أو من أجهزة أخرى في النظام، يقوم المعالج بقراءة هذه البيانات وتخزينها في السجلات الداخلية للمعالج.
4. تنفيذ التعليمة (Execute): يقوم المعالج بتنفيذ العملية المحددة في التعليمة باستخدام البيانات التي أحضرت من الخطوة السابقة.
5. تخزين النتيجة (Result Write): يقوم المعالج بتخزين النتيجة الناتجة عن تنفيذ التعليمة في الذاكرة أو في السجلات الداخلية للمعالج حسب ما يتطلبه النظام.

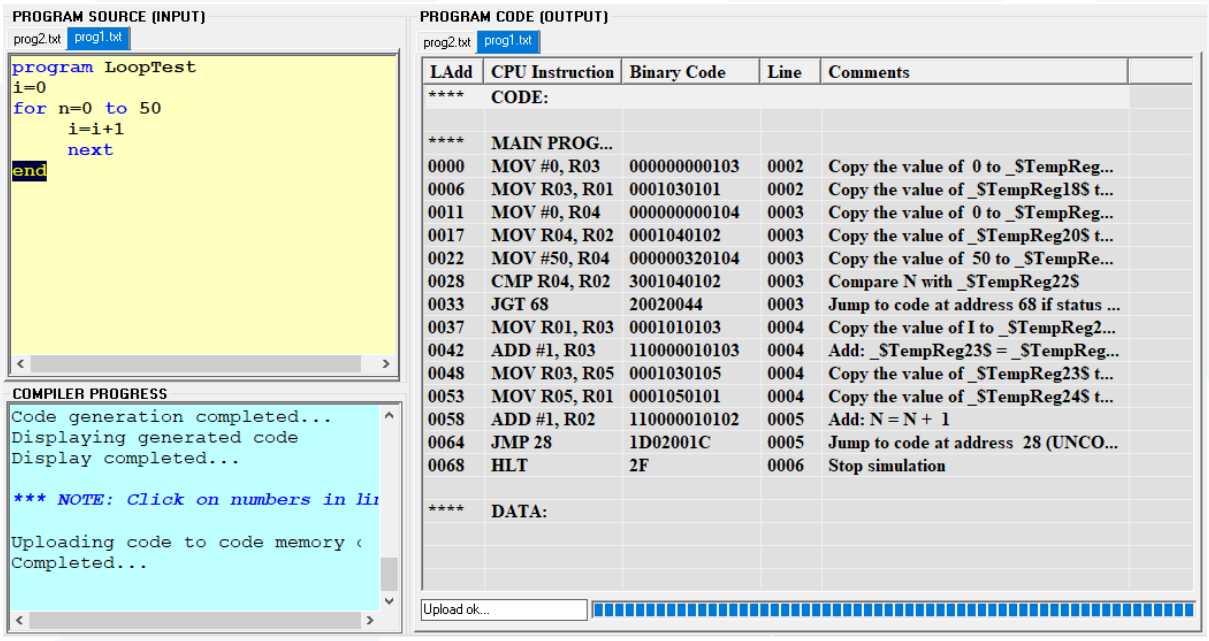
4. تمارين

تمرين 1:

نكتب في نافذة البرامج (Compiler) البرنامج التالي:

```
program LoopTest

i=0

for n=0 to 50

        i=i+1

        next

end
```

ثم نقوم بعمليتي complie و تحميل في الذاكرة ضمن عنوان أساسي ما:



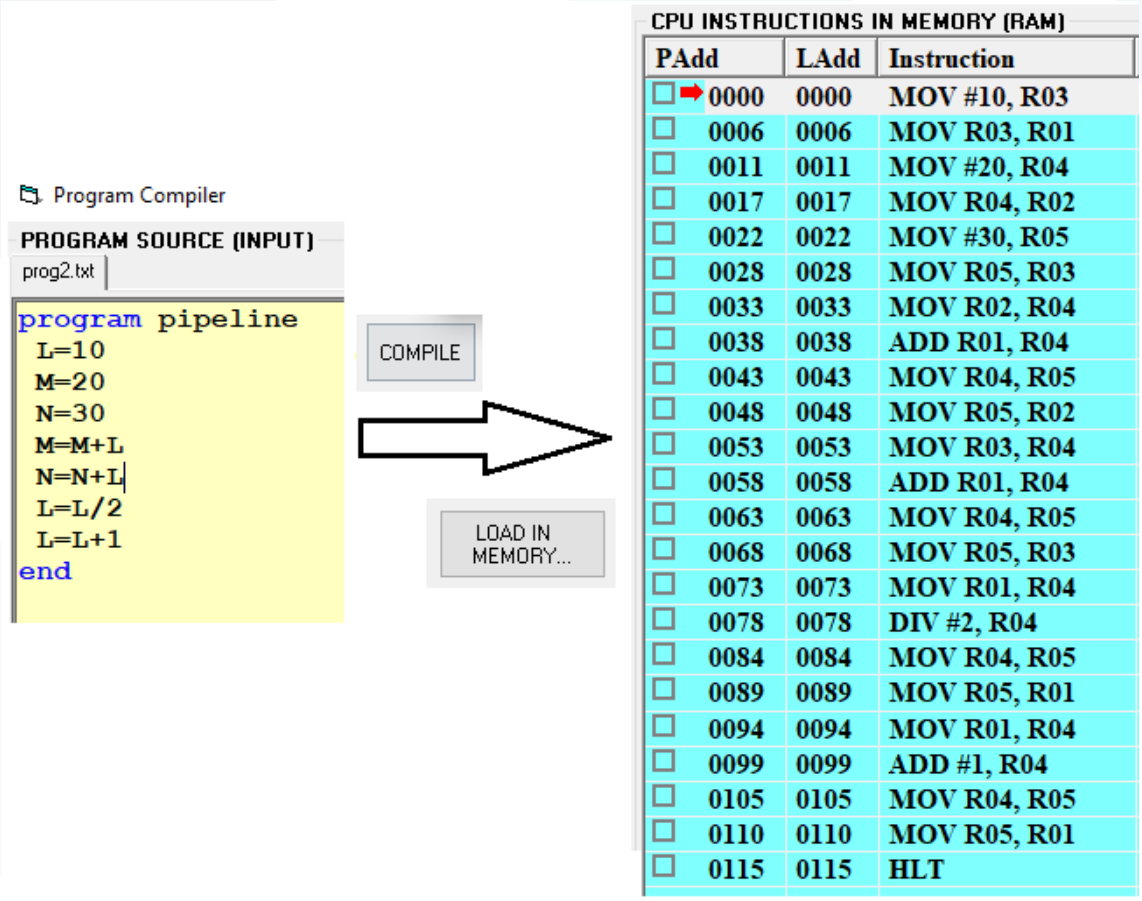
الشكل 3 – التمرين الأول بلغة عالية المستوى ولغة التجميع

تمرين 2:

نكتب في نافذة البرامج (Compiler) البرنامج التالي:

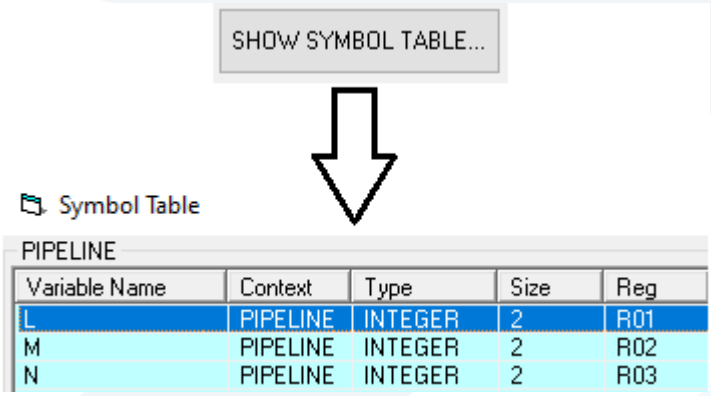
```
program pipeline
        L=10
        M=20
        N=30
        M=M+L
        N=N+L
        L=L/2
        L=L+1
end
```

ثم نقوم بعمليتي complie و تحميل في الذاكرة ضمن عنوان أساسي ما:



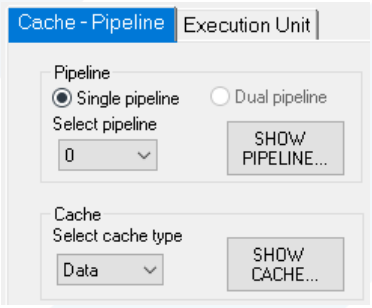
الشكل 4 – التمرين الثاني بلغة عالية المستوى ولغة التجميع

بعد تنفيذ البرنامج يمكن استعراض جدول المتغيرات الخاص به:

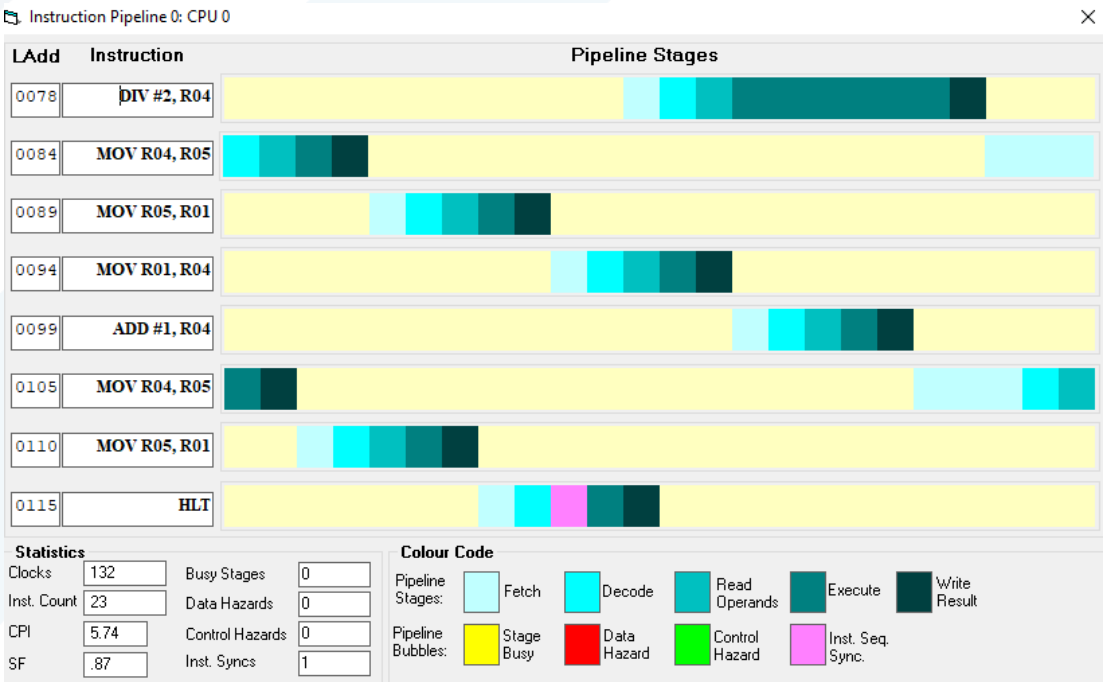


الشكل 5 – جدول المتغيرات الخاص بالتمرين الثاني

يمكن فتح نافذة عرض أنبوبة المعالجة Show Pipeline من النافذة الرئيسية لبرنامج المحاكي:

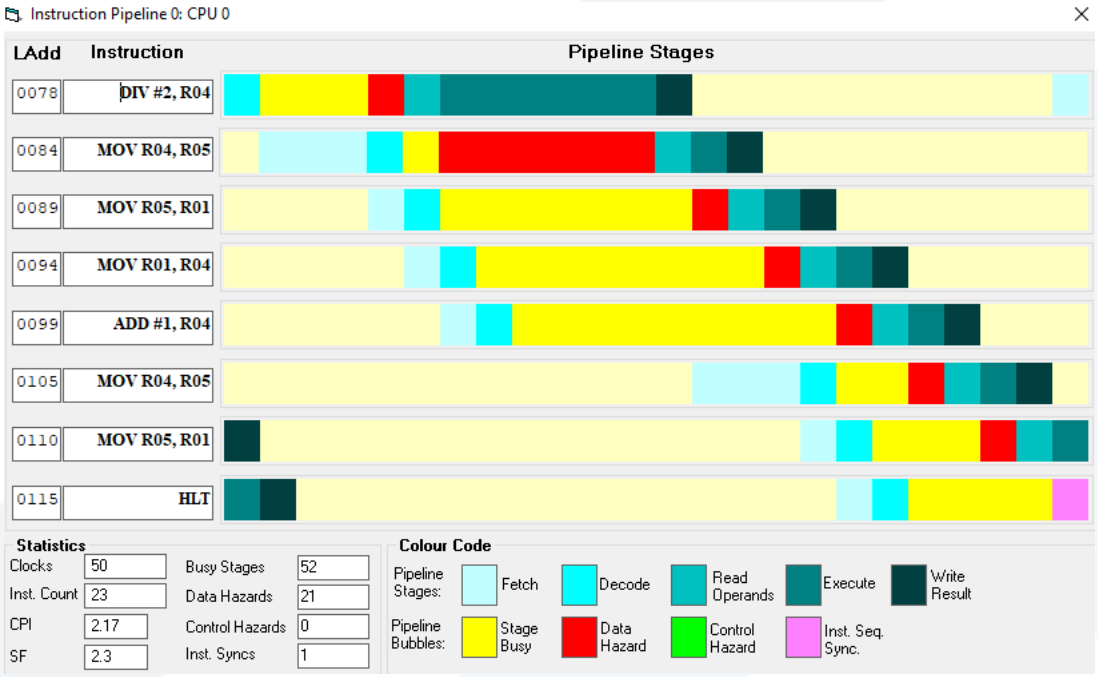


الشكل 6 – نافذة فتح أنبوبة المعالجة Pipeline



الشكل 7 – مخطط تنفيذ التعليمات دون تفعيل عملية أنبوبة المعالجة للتمرين الثاني

يمكن ملاحظة أن عملية أنبوبة المعالجة ليست فعالة بعد، المعالج ينتظر التعليمة حتى تنتهي بشكل كامل حتى يجلب التعليمة التالية ويترجمها وينفذها، وهو شي ء غير عملي حيث أن عدد دورات الساعة 132، معامل السرعة 0.87. لذلك يجب تفعيلها.



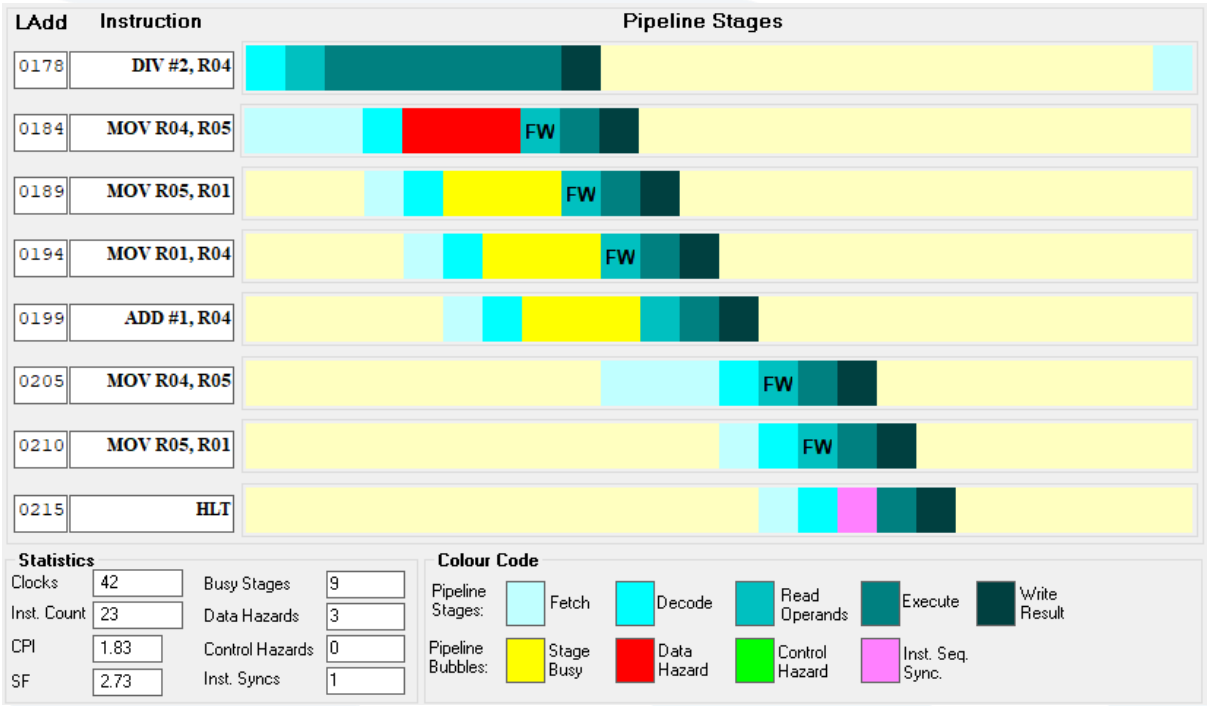
الشكل 8 – مخطط تنفيذ التعليمات مع تفعيل عملية التنبيب للتمرين الثاني

نلاحظ أن عدد دورات الساعة تناقص حتى 50 دورة فقط، وزاد معامل السرعة إلى 2.3,

لكن عدد مخاطر البيانات Data Hazards المبينة باللون الأحمر هو 21. ومخاطر البيانات هي إحدى أنواع المخاطر التي يجب أن تتعامل معها وحدة المعالجة المركزية أثناء تنفيذ التعليمات، وتحدث عند عدم توفر قيمة المعامل المطلوب لتنفيذ التعليمة.

إحدة الطرق المستخدمة لتقليل هذا العدد الكبير من الفقاعات Bubbles (اسم شائع للمخاطر) هي جعل وحدة المعالجة المركزية تسرّع من توفر المعاملات للتعليمات الموجودة في الأنابيب، وأهم هذه الطرق هي توجيه المعامل Operand Forwarding,

يمكن تفعيل خيار توجيه المعامل من نافذة أنبوبة المعالجة نفسها وإعادة تنفيذ البرنامج، فينتج لدينا:



الشكل 9 – مخطط تنفيذ التعليمات مع تفعيل عملية أنبوبة المعالجة وتوجيه المعاملات للتمرين الثاني

نلاحظ انخفاض عدد دورات الساعة وزيادة معامل السرعة إضافة إلى انعدام مخاطر البيانات (الفقاعات) تقريباً.