جَامعة
المَنارة
MANARA UNIVERSITY

Compilers Techniques
Lecture1
المحاضرة الأولى

Introduction to compilers

مدخل إلى المترجمات

Reference: Alfred, V. Aho, S. Lam Monica, and D. Ullman Jeffrey. "Compilers Principles, Techniques & Tools." (2007), *Pages 1-24*

السنة الرابعة –المستوى السابع- الهندسة المعلوماتية

مدخل إلى المترجمات

Introduction to compilers

الغاية من المقرر

لابد من ترجمة أي برنامج مكتوب بلغة برمجة ما، قبل أن يدخل مرحلة التنفيذ. والذي يقوم بهذه الترجمة عادةً هو نظام برمجي software system يطلق عليه اسم المترجم compiler.

يهدف هذا المقرر لتقديم المبادئ والتقنيات المستخدمة في عمليات الترجمة هذه، إضافةً إلى المفاهيم التي ستعرض لنا أثناء بناء المترجمات.

لغة الآلة Machine Languages (1st generation)

لغة التجميع Assembly:أوائل الخمسينيات (2nd generation).

اللغات عالية المستوى High-Level Languages:أواخر الخمسينيات (3rd generation) مثل C, C++.

لغات الجيل الرابع عالية المستوى 4th generation higher level languages مثل (SQL, Python, Matlab, Ruby, Perl)

لغات الجيل الخامس عالية المستوى 5th generation languages مثل (logic based, eg, Prolog)

أمثلة عن الأخطاء التي تكشفها المترجمات قبل التنفيذ:

نقص في بناء التعليمة

إضافة أجزاء لا تصلح وفقاً لقواعد اللغة.

تكرار التصريح عن متحولات

استخدام متحولات غير مصرح عنها

أخطاء في أنواع البيانات

الخ ...

input

Compilation failed due to following error(s).

```
main.c: In function ‘main’:
main.c:15:6: error: expected expression before ‘.’ token
   if(.)
      ^
```

```
try.c: In function 'main':
try.c:5:9: error: redefinition of 'a'
     int a = 7;
         ^
try.c:4:9: note: previous definition of 'a' was here
     int a = 5;
```

المخرجات التعليمية:

بانتهاء الطالب من دراسة هذا المقرر يجب عليه:

فهم المبادئ التي تحكم مراحل وأطوار عملية الترجمة.

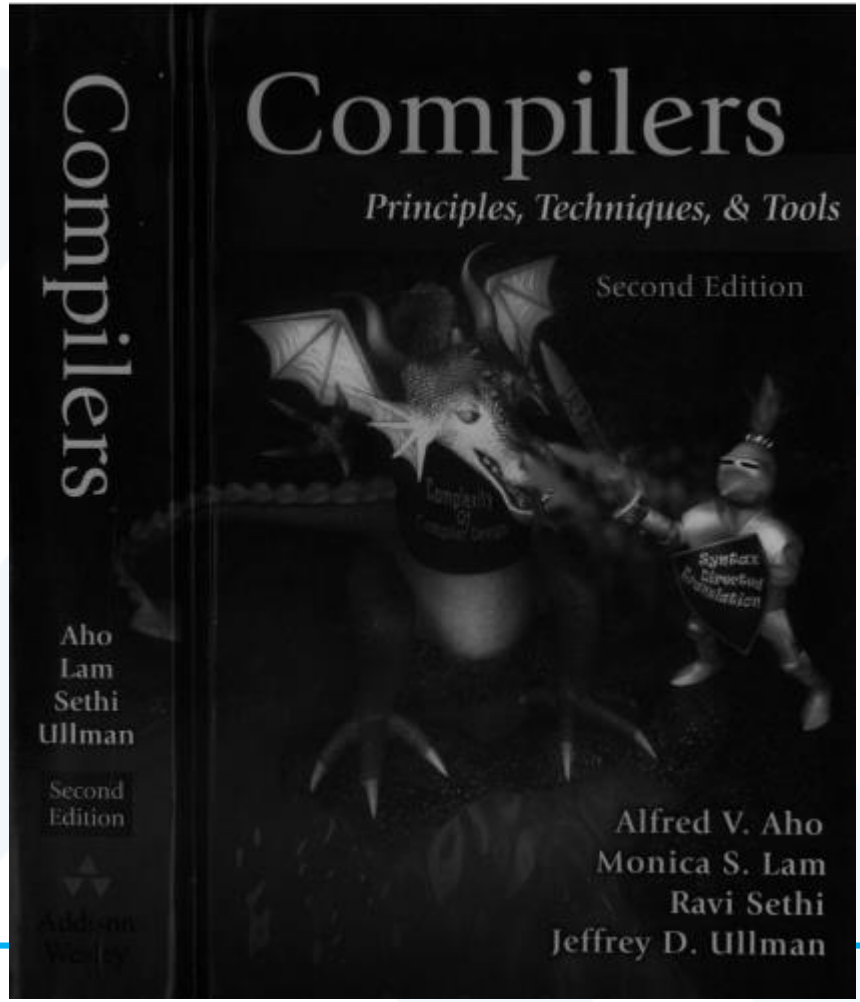
فهم مبدأ عمل ودور كل مرحلة من مراحل الترجمة.

فهم مبدأ عمل وكيفية تطبيق التقنيات الأساسية في كل مرحلة من مراحل الترجمة.

إمكانية بناء مترجم بسيط للغة مصدرية ما (على الأقل المراحل الأساسية للمترجم).

الكتاب المرجعي

Alfred, V. Aho, S. Lam Monica, and D. Ullman Jeffrey. "Compilers Principles, Techniques & Tools." (2007).

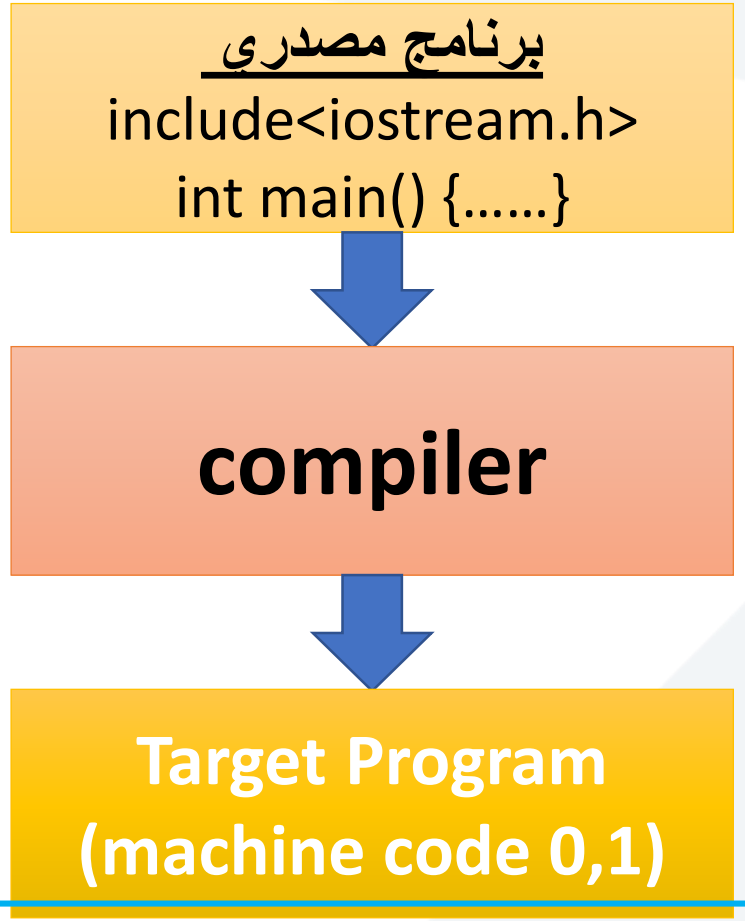


المفردات Syllabus

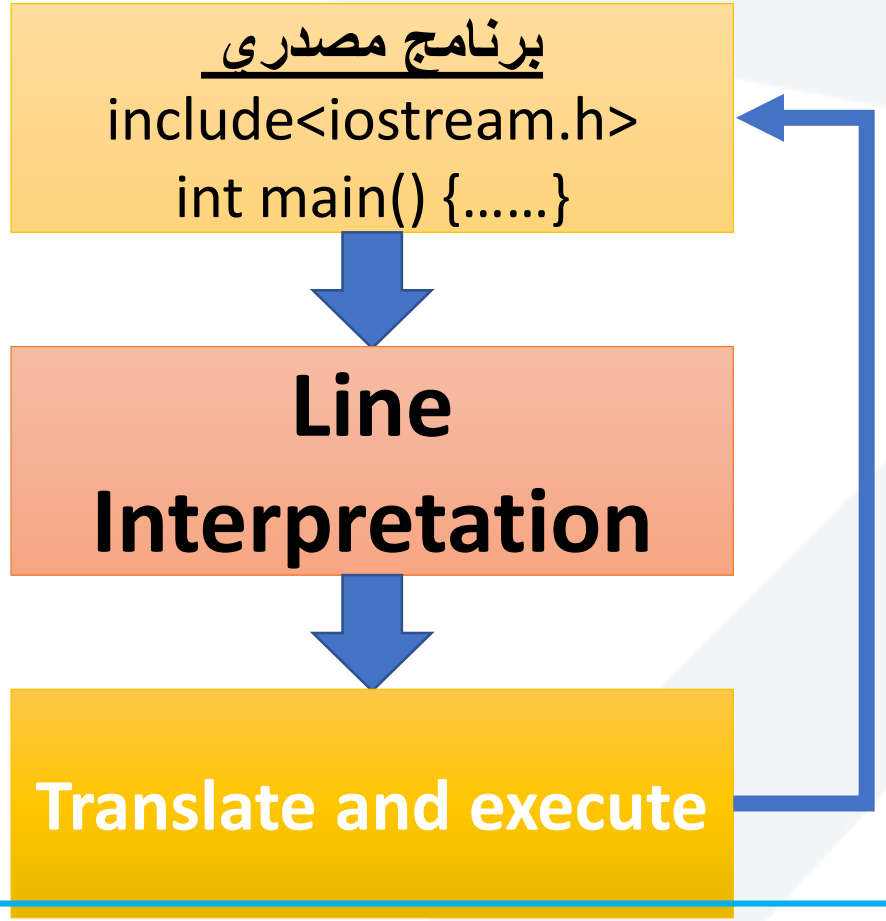
1- مدخل إلى المقرر Introduction
2- التحليل المعجمي Lexical Analysis (scanning)
3- التحليل القواعدي Syntax Analysis (parsing)
4- التحليل الدلالي Semantic Analysis
5- التمثيل الوسيطي Intermediate Representations
6- إدارة التخزين Storage Management
7- توليد الشيفرة Code Generation
8- أمثلة الشيفرة Code Optimisation

تعاريف Definitions

- ما هو المترجم ?What is a compiler
  - برنامج يقبل كدخل نص البرنامج مكتوباً بلغة ما وينتج كخرج نص البرنامج مكتوباً بلغة أخرى مع المحافظة على مايعنيه النص..(Grune *et al*, 2000)
  - برنامج يقرأ برنامجاً مكتوباً بلغة ما (اللغة المصدر source language) ويترجمه إلى برنامج مكافئ بلغة أخرى (اللغة الهدف target language) (Aho *et al*)
- يقرأ المترجم البرنامج ويتحقق منه إذاً قبل تنفيذ أي جزء منه وفي حال كشفه لأي خطأ يظهر رسالة خطأ تحتوي ماهية الخطأ ورقم السطر الذي يتضمن الخطأ.
- في حال خلو البرنامج من الأخطاء يتم تنفيذ البرنامج (تحويله إلى كود قابل للتنفيذ Executable Code).



تعاريف Definitions



ما هو المفسر Interpreter؟

برنامج يقرأ البرنامج المصدري سطراً-سطراً ويتحقق منه ثم يقوم بتنفيذ هذا السطر.

يتابع بعدها للسطر التالي ويكرر العملية.

لا ينتج المفسر شيفرة وسيطة Intermediate Code على عكس المترجم.

يتطلب ذاكرة أقل.

يظهر خطأ واحد فقط (سطر –سطر)

أسهل في التصميم والتعامل

أبطأ من المترجم

تعاريف Definitions

- C و C++ تترجم compiled
- Perl, PhP, Python تفسر interpreted
- Java تترجم إلى bytecodes، بعدها تُفسر interpreted.

أيضاً:

- C++ to Assembly
- C++ to C
- High Performance Fortran (HPF) to Fortran (parallelising compiler)
- C to C (or any language to itself)

يجب على المترجم أن:

- 1- يحافظ على دلالة البرنامج الذي يترجمه (لا يتغير الهدف)
- 2- يحسّن الشيفرة المصدرية بشكل أو بآخر.

Why to study compilers

لماذا ندرس المترجمات

لنتعرف على كيفية تنفيذ البرامج وفقاً للغات البرمجة المختلفة.

لنتعلم كيفية قراءة الأخطاء القواعدية والمعنوية التي تحدث أثناء تنفيذ البرامج.

لنتعلم كيفية التفاعل بين البرامج التي نكتبها وبين الحاسب الذي يقوم بتنفيذها.

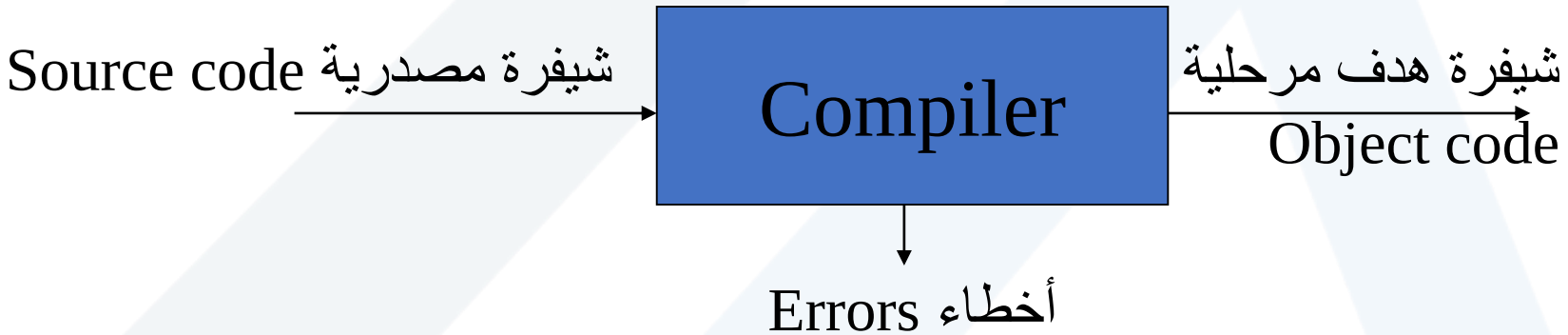
لنتعلم كيفية تنقيح الأخطاء في الشيفرة (الكود) وكيفية تحسين الشيفرة لتنفذ بشكل فعال أكثر على موارد الحاسب (بأقل استهلاك لموارد الحاسب).

General Compiler Architecture

بنية المترجم العامة

يجب على المترجم أن:

- يحلل الشيفرة المصدرية analyse
- يتعرف على الأخطاء Debug
- ويركب الشيفرة الوسيطية synthesis ويحسنها Optimize
- ثم يولد الشيفرة النهائية الصحيحة Generate Final Code.



- General Compiler Architecture

بنية المترجم العامة
البنيتين الأساسيتين للمترجم

- **يتألف المترجم من جزأين أساسيين:**
- Front-End ودخله هو الشيفرة المصدرية Source Code أما خرجه فهو التمثيل الوسيطي Intermediate code representation (كود وسيط خالي من الأخطاء المعجمية والقواعدية والمعنوية مع بناء شجرة الإعراب).
- Back-End دخله هو التمثيل الوسيطي أما خرجه فهو شيفرة الهدف Target Code.

بنية المترجم العامة
البنيتين الأساسيتين للمترجم

- General Compiler Architecture

- **ينجز الطرف الأمامي Front-end** تحليل اللغة المصدرية source language.
  - يميز البرامج الصحيحة من تلك التي تتضمن أخطاء ويصدر تقريراً بالأخطاء.
  - يفهم برنامج الدخل (البرنامج المصدري) ويجمع عناصره الدلالية في تمثيل وسيطي.
  - يولد تمثيلاً وسيطياً ويشكل الشيفرة بما يتناسب مع Back-End.
  - يمكن أتمتة عمل الكثير مما ذكر.

بنية المترجم العامة
البنيتين الأساسيتين للمترجم

- **الطرف الخلفي Back-end**: يقوم بتركيب اللغة الهدف target language.
  - يختار التعليمات لتنفيذ كل عملية من عمليات التمثيل الوسيطي.
  - يفسر Translates التمثيل الوسيطي IR إلى شيفرة هدف target code.
  - احتمال نجاح الأتمتة أقل من الطرف الأمامي.
- درجة تعقيد الطرف الأمامي هو **O(n)** حيث n حجم الكود المصدري،
- بينما درجة تعقيد الطرف الخلفي هو

NP-complete.

A problem which we don't know how to solve in less than exponential time

وينتج عن الفصل بين الطرف الأمامي (التحليل) والطرف الخلفي (التركيب) لغة جديدة.

البنية العامة للمترجم