處理器設計與實作

實習講義

編撰者
成大電通所計算機架構與系統研究室CASLAB

國立成功大學電機系與電腦與通信工程研究所

LAB 5: RISC-V & LLVM
RISC-V編譯器實作

大綱

1. Compiler 基本概念
2. LLVM 基本概念
3. 實驗: RISC-V LLVM實作
   - 練習題一
   - 挑戰題

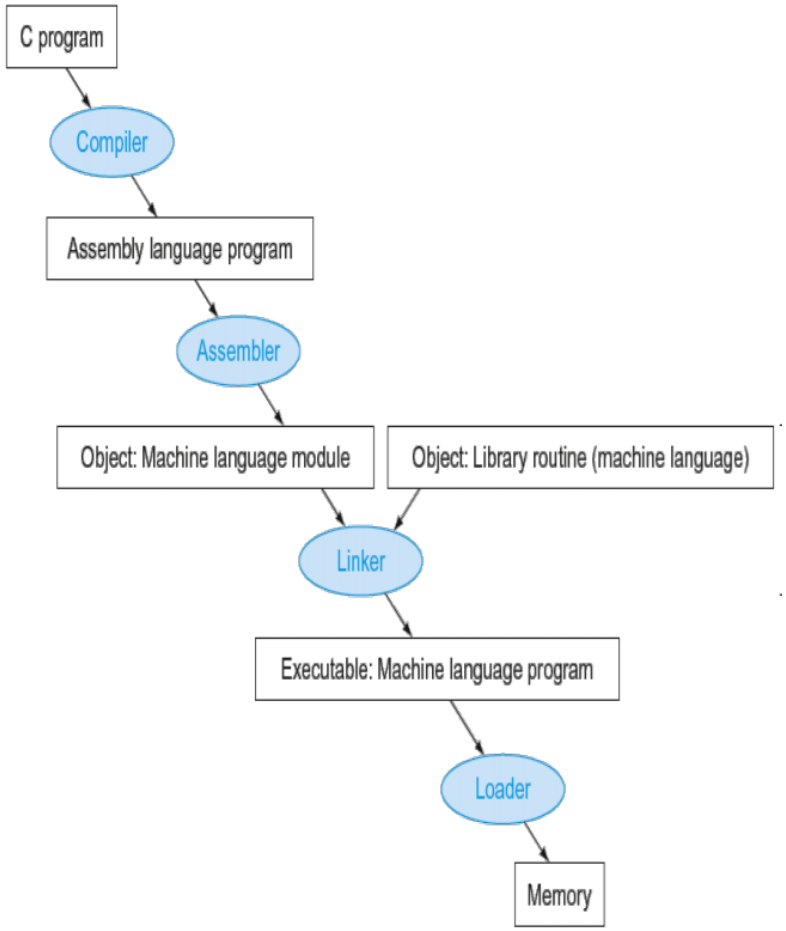
實驗目的

1. 認識編譯器與程式轉換流程
2. 嘗試在編譯器中新增客製化指令並與硬體結合
3. 認識編譯器的優化

Compiler 基本概念

What is compiler?

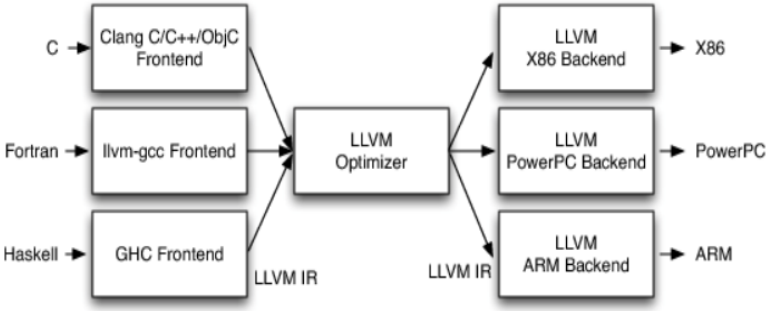
- Translator between engineer & machine
- Make programming easier
- Optimize your code

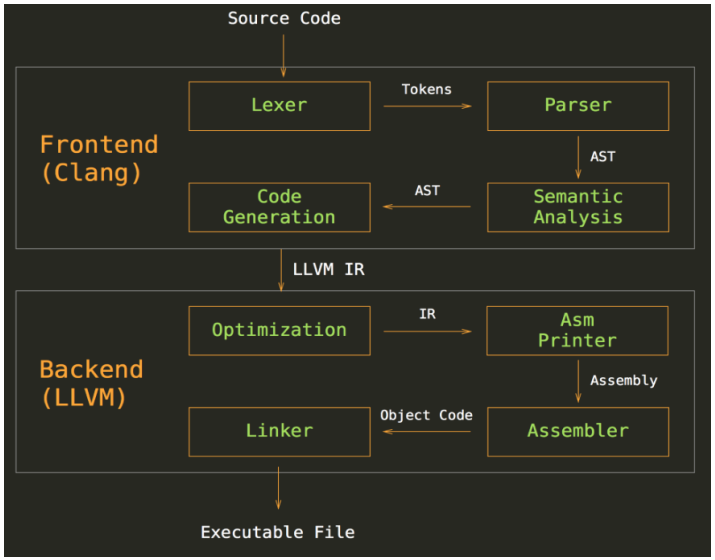


LLVM 基本概念

What is LLVM

- Begin as a research at the University of Illinois
- Modular and reusable compiler and toolchain technologies
- Provide common and simple LLVM IR
- Easier to target new SW and HW





LAB 5: RISC-V & LLVM

Tool used

實驗環境:

1.Linux

- LLVM with RISC-V backend
- RISC-V GNU toolchain

2. Windows

- Modelsim

Lab 5-1 : Custom RISC-V instruction

Complete instruction definition:

- ~/Workspace/Tools/llvm-project/llvm/lib/Target/RISCV/
  - RISCVInstrFormats.td
  - RISCVInstrInfo.td

Instruction	Funct7	rs2	rs1	xd/xs1/xs2	rd	opcode
mac.load	1	0	GPR	011	0	0001011
mac.rfc0	2	0	0	111	GPR	0001011

```
//====================Custom Instruction====================\\

def OPC_CUSTOM     : RISCVOpcode<0b>;
class RVInstR_CUSTOM<bits<7> funct7, bits<3> funct3, RISCVOpcod
              dag ins, string opcodestr, string argstr>
    : RVInst<outs, ins, opcodestr, argstr, [], InstFormatR> {
  bits<5> rs2;
  bits<5> rs1;
  bits<5> rd;

  let Ins{-} = funct7;
  let Ins{-} = rs2;
  let Ins{-} = rs1;
  let Ins{-} = funct3;
  let Ins{-} = rd;
  let Opcode = opcode.Value;
}
//============================================================\\
```

RISCVInstrFormats.td

```
//===============Custom Instruction====================\\

let hasSideEffects = 0, mayLoad = 0, mayStore = 0 in
class CUSTOM_INSTR_1<bits<7> funct7, bits<3> funct3, string opcodestr>
    : RVInstR_CUSTOM<funct7, funct3, OPC_CUSTOM, (outs), (ins GPR:$),
              opcodestr, "$">;

let hasSideEffects = 0, mayLoad = 0, mayStore = 0 in
class CUSTOM_INSTR_2<bits<7> funct7, bits<3> funct3, string opcodestr>
    : RVInstR_CUSTOM<funct7, funct3, OPC_CUSTOM, (outs GPR:$), (ins),
              opcodestr, "$">;

def MLD  : CUSTOM_INSTR_1<0b, 0b, "custom1">, Sched<[]>{
  let ;
  let ;
}
def MRF  : CUSTOM_INSTR_2<0b, 0b, "custom2">, Sched<[]>{
  let ;
  let ;
}
//=====================================================\\
```

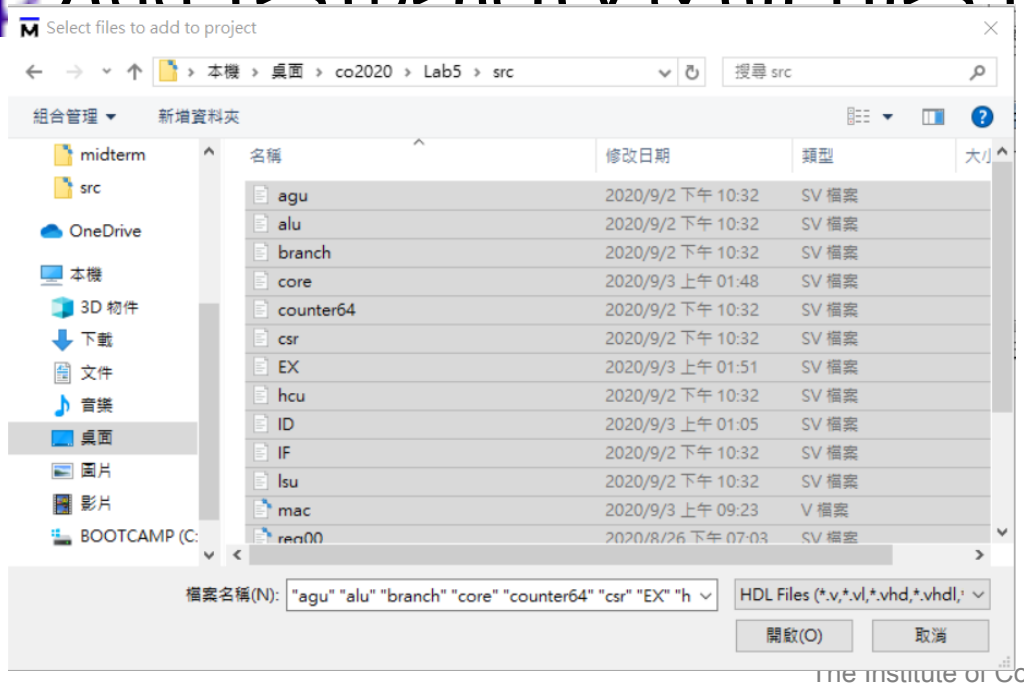
RISCVInstrInfo.td

Lab 5-1 : Custom RISC-V instruction

- Re-build LLVM compiler
  - cd ~/Workspace/Tools/llvm-project/build
  - sudo cmake –build . –target install
- Generate memory file
  - cp test.c link.ld bin2mem.py ~/Workspace
  - cd ~/Workspace && sh compile.sh

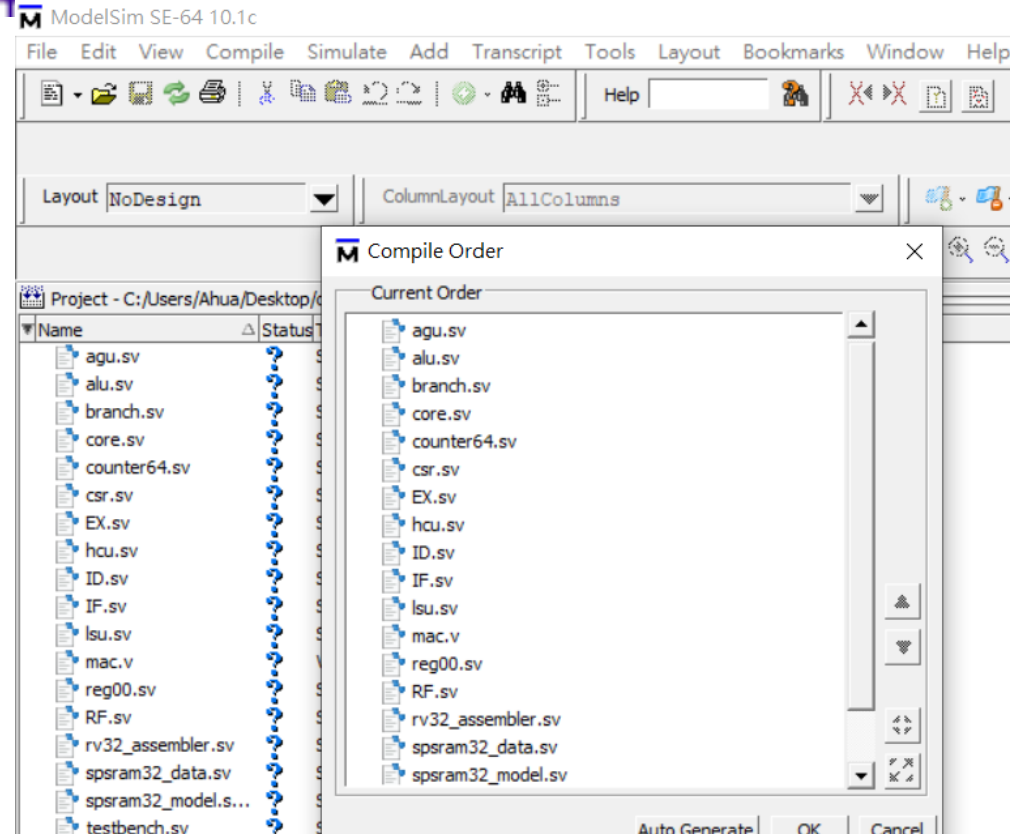
Modelsim驗證

- Copy test.mem to windows under Lab5/
- Open Modelsim and create new project under Lab5/
- Add testbench.v & all files in src/*



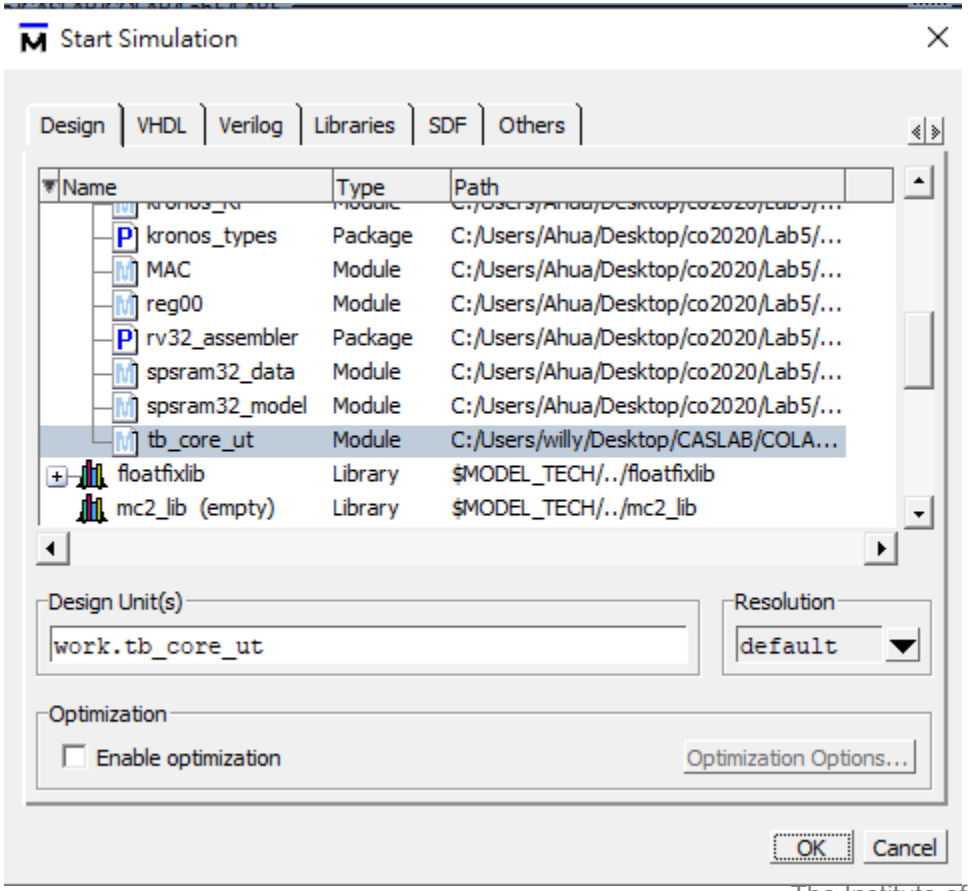
Modelsim驗證

- Compile -> Compile order
- Auto Generate

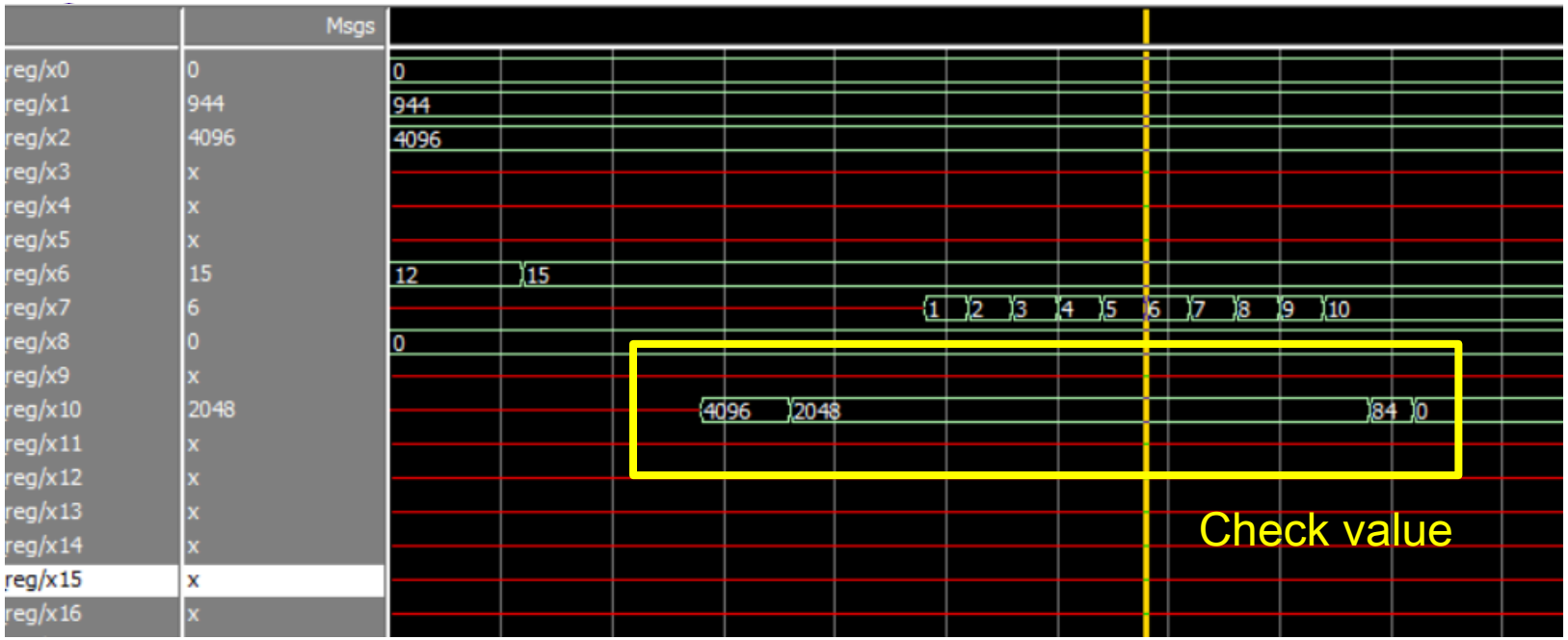


Modelsim驗證

- Simulation 選擇 work -> tb_core_ut
- Disable optimization



預期結果


Msgs
reg/x0
reg/x1
reg/x2
reg/x3
reg/x4
reg/x5
reg/x6
reg/x7
reg/x8
reg/x9
reg/x10
reg/x11
reg/x12
reg/x13
reg/x14
reg/x15
reg/x16
0
944
4096
x
x
x
15
6
0
x
2048
x
x
x
x
x
x
0
944
4096
12
15
1
2
3
4
5
6
7
8
9
10
0
4096
2048
84
0
Check value

Challenge 挑戰題

- Compile source with/without optimization
  - ➢ cp origin.c ~/Workspace
  - ➢ /opt/RISCV/LLVM/bin/clang -S -emit-llvm -Xclang -disable-O0-optnone origin.c
  - ➢ /opt/RISCV/LLVM/bin/opt -mem2reg -S origin.ll -o opt.ll
- Compare difference between origin.ll & opt.ll

實驗結報

- 結報格式(每組一份)
  - 封面 (第幾組+組員)
  - 實驗內容(程式碼註解、結果截圖)
  - 實驗心得
- 繳交位置
  - ftp://140.116.164.225/ port: 21
  - 帳號/密碼：ca_lab/Carch2020
  - Deadline: 11/09 18:00pm
- TA Contact Information:
  - 助教信箱：anita19961013@gmail.com
  - Lab：92617
  - Office hour：(Tuesday)8:00pm~10:00pm