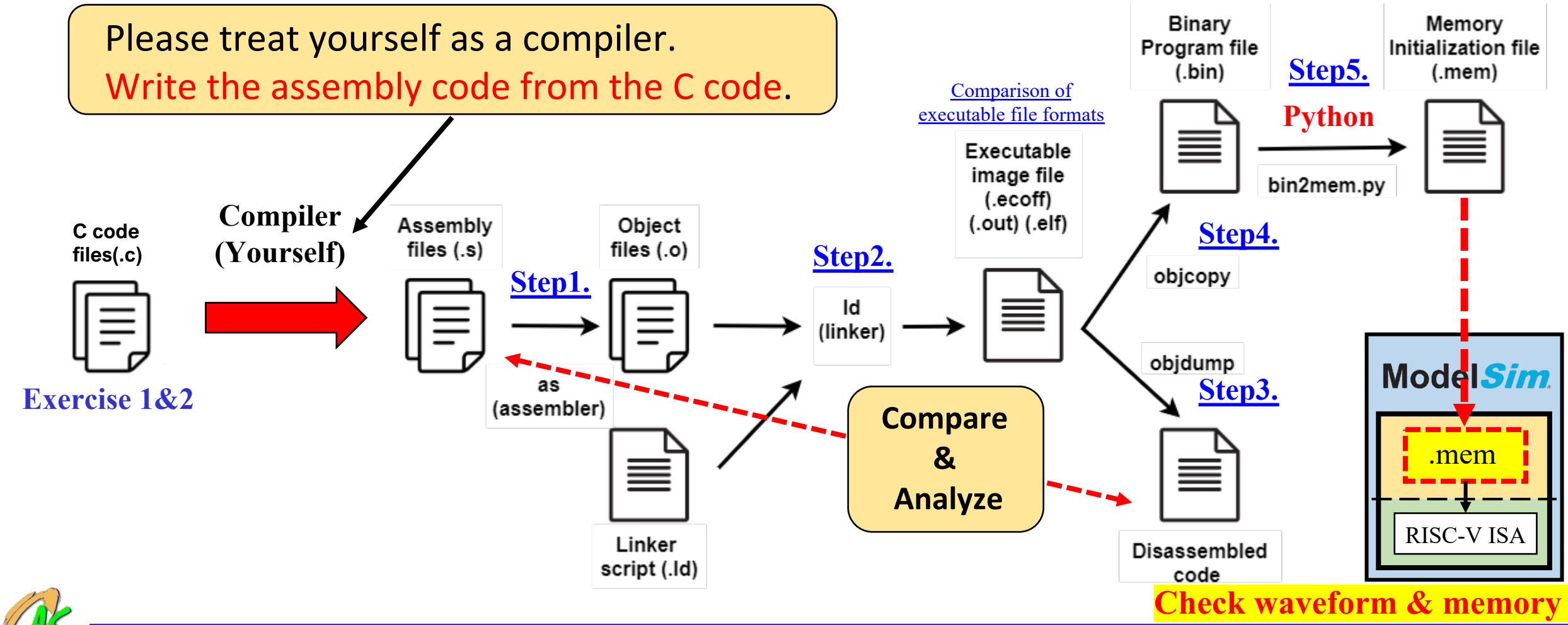
Lab 2
Assembly Lab I
Exercise & Report Format

Video Link : https://youtu.be/pO-5Ls6AdXo

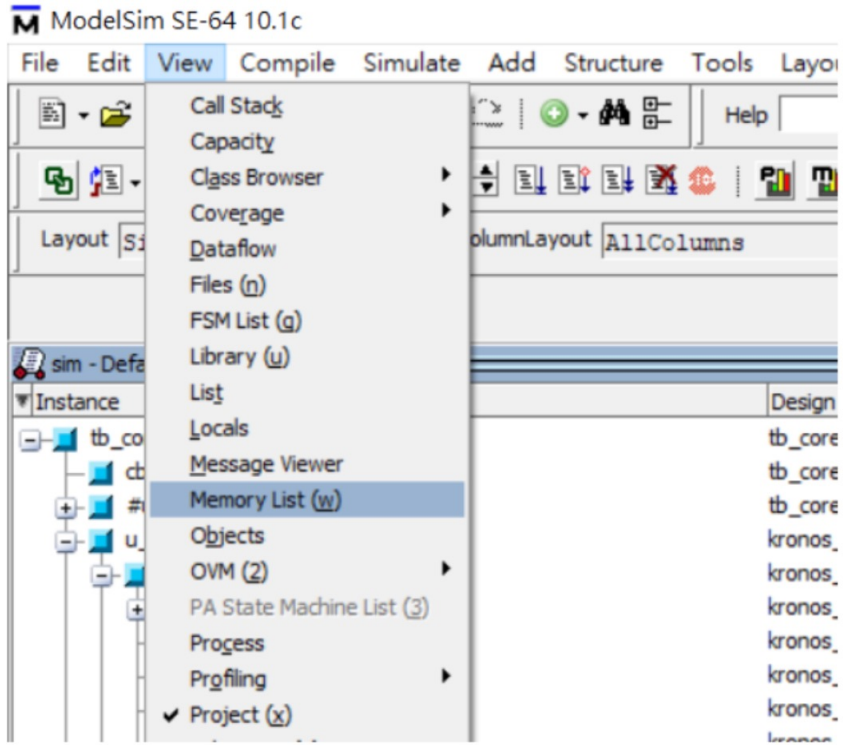
Exercise Workflow

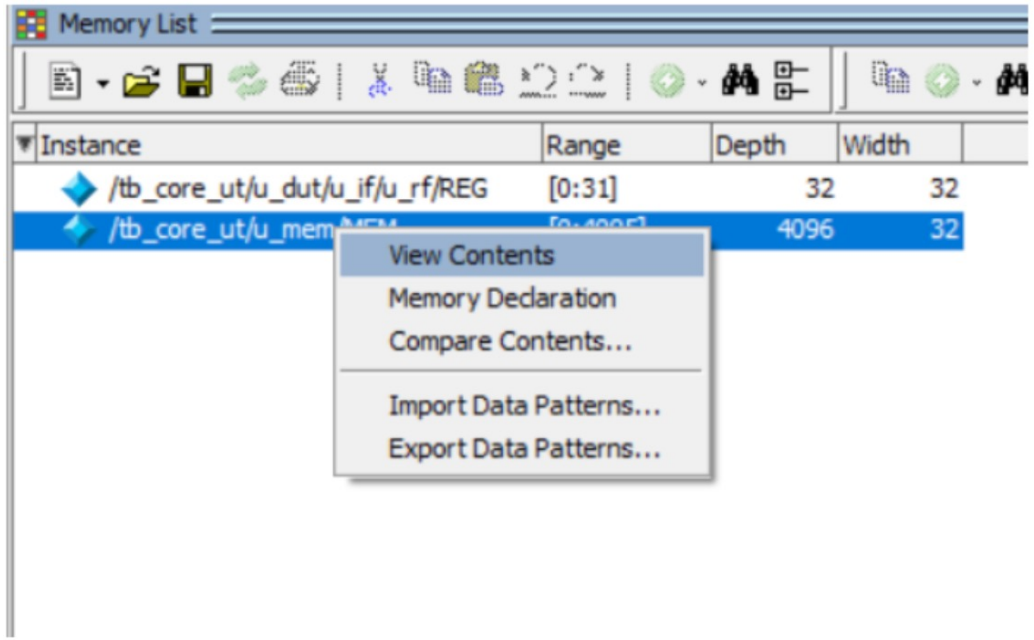
Please treat yourself as a compiler.
Write the assembly code from the C code.



ModelSim – Check Memory

1. View -> Memory List
2. Right Click /tb_core_ut/u_mem/MEM -> View Contents

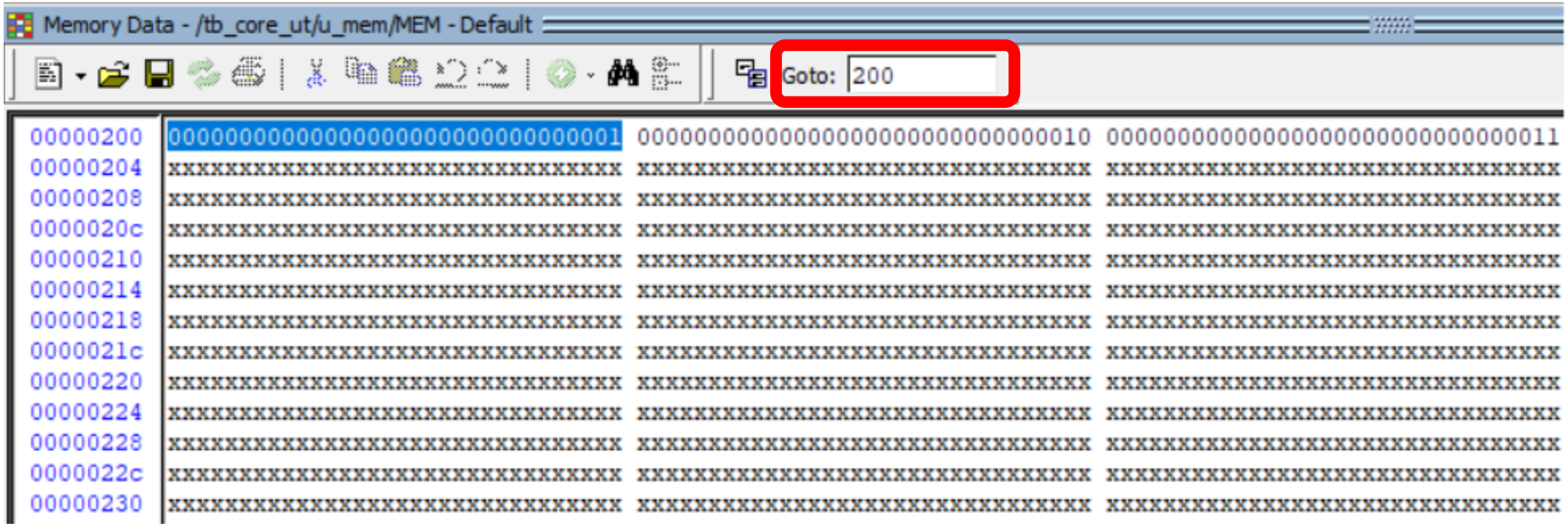




ModelSim – Check Memory

3. **Goto** : The memory **word address** you want to check
   (**View -> Properties**, we can change radix form here)

The memory address in ModelSim is word address.
Byte address : 12 => Word address : 3
Byte address : 1024 => Word address : 256



Exercise 1 (Lab2/Exercise1)

- Please write the RISC-V assembly code (ex1.s) to implement the C code and save the final sum execution result to $t1.
- Please use Toolchain & Python to convert “ex1.s” you wrote to “ex1.mem” file
- Please disassemble (objdump) the “ex1.elf” to “ex1.dump” and compare the difference between “ex1.s” and “ex1.dump” and explain in report.
- Please use ModelSim to simulate and check the waveform of registers
- Screenshot the waveform & explain it in report with the “ex1.dump”

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
    int a = 10;
    int b = -16;
    int c = -40;
    int d = a%4;
    int sum = 0;
    sum += b*8 + a/7 + (b/4 + 66 - 30)*9 + a*3.5 + abs(c) - d;
    return 0;
}
```

a/8

Question List

Q1 :
Why IALIGN of RV32I is 32 bits ?
Why IALIGN need to support 16 bits ?
Why IALIGN have no greater than 32 bits (e.g. 64 / 128 bits) ?

Q4 : Why there is no lwu in RV32I ?

Q5 : What is the addressing mode of Load & Store ?

Q2 : Why temporary registers and saved registers are not numbered sequentially ?

# Registers	Base	Version	Status	
	RVWMO	2.0	**Ratified**	Weak Memory Ordering
32	**RV32I**	**2.1**	**Ratified**	Base Integer Instruction Set, 32-bit
32	**RV64I**	**2.1**	**Ratified**	Base Integer Instruction Set, 64-bit
16	*RV32E*	*1.9*	*Draft*	Base Integer Instruction Set (embedded), 32-bit
32	*RV128I*	*1.7*	*Draft*	Base Integer Instruction Set, 128-bit

```
int  abc(int a, char b, int* c)
 ↑      ↑          ↑
a0 ~ a1          a0 ~ a7
```

Q3 : Why return value needs 2 registers (a0, a1) ?

	int	long	pointer
ilp32/ilp32f/ilp32d	32-bit	32-bit	32-bit
lp64/lp64f/lp64d	32-bit	64-bit	64-bit

RISC-V 只支援這六種

LLP = long long & pointer

	ILP32	LP64	LLP64	ILP64
char	8	8	8	8
short	16	16	16	16
int	32	32	32	64
long	32	64	32	64
long long	64	64	64	64
void *	32	64	64	64