Introduction to MIPS programs

COMP2611:
Computer Organization

❑ You will learn the following in this tutorial:

    ❑ MIPS data types and some pseudoinstructions.

    ❑ the translation of program codes between MIPS and C++.

MIPS

MIPS data types
- the definition of MIPS data types
Program codes
- text segment
- pseudoinstructions
- exercises
Exercises

We need to declare "variables" & "Arrays" used in the program in a data segment.

- The compiler recognizes .data as the beginning of data segment

.data

h: .word 1 2 3 4   # h is an array of size 4, each element is a word (32 bit)

s: .word 5

The 3 lines below let the system know the program begins here

.text

.globl __start

__start:

Write your program code here

la $s0, h   # Obtain starting address of array h, s0 = x (a constant)

lw $s1,8($s0)   # $s1 = content in memory address x + 8 = 3 = h[2]

la $s2, s

lw $s3, -12($s2)

sub $s3, $s3, $s1

sw $s3, 0($s0)

- ❑ .ascii
  - ❑ String (without null terminator)

- ❑ .asciiz
  - ❑ String (with null terminator)
  - ❑ "z" refers to zero (ASCII code for the null character)

- ❑ .byte
  - ❑ Byte
  - ❑ values can be written in either base 10 or in hex

❑ .half

    ❑ 2 bytes

    ❑ values can be written in either base 10 or in hex

    ❑ Half word aligned in memory

    ❑ i.e. initial byte stored at addresses divisible by 2

❑ .word

    ❑ 4 bytes

    ❑ values can be written in either base 10 or in hex

    ❑ word-aligned in memory

    ❑ i.e. initial byte stored at addresses divisible by 4

❑ .space num

    ❑ Reserves num bytes of space in memory.

Variable name	Data type	Initialized value	Remarks
var1:	.half	14	# A half-word storing the integer 14
array1:	.word	5 6 7 8	# same as int array1[4] = {5,6,7,8} in C++
array2:	.word	3:5	# the part before ":" in the initialized value is # the initial value of each element in the # array, and the part after ":" is the array size. # same as int array2[5] = {3,3,3,3,3} in C++
string1:	.byte	0x32 # '2' in ASCII code 0x4a # 'J' in ASCII code 0 # '\0' in ASCII code	# string type is actually an array of char (a byte) # same as char string1[3] = {'2','J','\0'} in C++
string2:	.asciiz	"2J"	# equivalent to string1
array3:	.space	10	An array of 10 bytes is allocated for array3 in memory.

MIPS

MIPS data types

- the definition of MIPS data types

Program codes

- text segment

- pseudoinstructions

- exercises

Exercises

❑ Under the .text segment of a MIPS program, we usually define the label __start or main for the first instruction of the program code in execution, like:

.text
.globl __start
__start:

❑ This label is often defined as a global label using .globl. A global label is a label that can be accessed across all the code files in a multiple-file assembly program. But you don't need to know this kind of program since we will not teach it.

❑ Defining this label or any others (using .globl or not) for the first instruction is optional.

High-level
language
program
(in C)

```
swap(int v[], int k)
{int temp;
    temp = v[k];
    v[k] = v[k+1];
    v[k+1] = temp;
}
```

Compiler

Assembly
language
program
(for MIPS)

```
swap:
        muli  $2,  $5,4
        add   $2,  $4,$2
        lw    $15,  0($2)
        lw    $16,  4($2)
        sw    $16,  0($2)
        sw    $15,  4($2)
        jr    $31
```

Assembler

Binary machine
language
program
(for MIPS)

```
00000000101000010000000000011000
00000000000110000001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

❑ They are MIPS assembly instructions that do not have the corresponding machine instructions. Some examples are:

Instructions	Explanation	Example	C++ Equivalent
(Load Immediate) **li Rdest, Imm**	Move the immediate *Imm* (constant) into register *Rdest*.	li $v0, 5	v0 = 5;
(move) **move Rdest, Rsrc1**	Copy the value of *Rsrc1* into register *Rdest*.	move $a0, $t0	a0 = t0;

Pseudoinstructions	Translation	Example
(Load Immediate) **li Rdest, small (16-bits)**	addi Rdest, Rdest, small	addi $t0, $t0, 0x0123
(Load Immediate) **li Rdest, big** (32-bits) e.g. 0x0123abcd	lui Rdest, upper(big) ori Rdest, Rdest, lower(big)	lui $r1, 0x0123 ori $r1, $r1, 0xabcd
(move) **move Rdest, Rsrc1**	addi Rdest, Rsrc1, 0	addi $t0, $s0, 0

Question 1: Write down the shortest sequence (any one) of MIPS instructions for the following C++ code, assuming each variable is stored in a different register (you name it):

a = a - 1;

Question 2: Write down the shortest sequence (any one) of MIPS instructions for the following C++ code, assuming each variable is stored in a different register (you name it):

b = a * 5;

Question 3: Write down the shortest sequence (any one) of MIPS instructions for the following C++ code, assuming the base address of the int array A is stored in the register s0. You can use some registers for storing temporary values.

A[2] = A[7] + 11;

Question 4: Write down the MIPS instructions for the following C++ codes, assuming each variable is stored in a different register (you name it). You can use some registers for storing temporary values.

```
if (d < 4) {
    if (d == 1)
        d = d + 4;
    else ++d;
}
```

MIPS

MIPS data types
- the definition of MIPS data types
Program codes
- text segment
- pseudoinstructions
- exercises
Exercises

Question 1: Write down the shortest sequence (any one) of MIPS instructions for the following C++ code, assuming the base address of the array A of int elements is stored in the register s0 and each variable is stored in a different register (you name it). You can use some registers for storing temporary values.

A[c++] = A[b] + 17;

Question 2: Write down the MIPS instructions for the following C++ codes, assuming each variable is stored in a different register (you name it). You can use some registers for storing temporary values.

```
switch (d) {
  case 1: d = d + 4;
          break;
  case 4: d = d * 2;
          break;
  default: d--;
}
```

❑ You have learnt:

    ❑   MIPS data types and some pseudoinstructions.

    ❑   the translation of program codes between MIPS and C++.