#### **Assignment 4 - Code Generation**

#### A few words about the MIPS architecture

Compilerconstructie

LIACS Universiteit Leiden

1

### The MIPS Machine

Highlights:

- Word-addressable (word = 4 bytes)
- Has 32 general purpose registers: **\$0, \$1, ..., \$31**
- Each register also has a name, e.g.: \$29 = \$sp (stack pointer)
- Most instructions have the following form:

opcode destination, source1, source2

## **MIPS** Registers

Provides 32 registers, but some are reserved or have a special meaning:

- **\$0:** Always 0
- **\$1:** Reserved for assembler
- **\$2:** Function return value
- **\$26-28:** Reserved for OS etc.
- **\$29:** Stack pointer
- **\$30:** Frame pointer
- **\$31:** Return address

#### **MIPS** Instructions

Examples of MIPS instructions:

add	\$2,	\$3,	\$4
sub	\$4,	\$4,	\$3
li	\$4,	123	
addi	\$4,	\$0,	123
beq	\$2,	\$3,	label
Iw	\$4,	X	

#### **MIPS Addressing Modes**

- MIPS is a load/store architecture: memory can be accessed only by load and store instructions.
- Data must be aligned properly.
- Main addressing modes:
  - Iw \$3, (\$2) Load memory (\$2) into register \$3
  - **Iw \$3, 4(\$2)** Load *memory*(**4+\$2**) into register **\$3**
  - Iw \$3, 200 Load *memory*(200) into register \$3
  - Iw \$3, x Load value of x into register \$3

## MIPS - Floating point

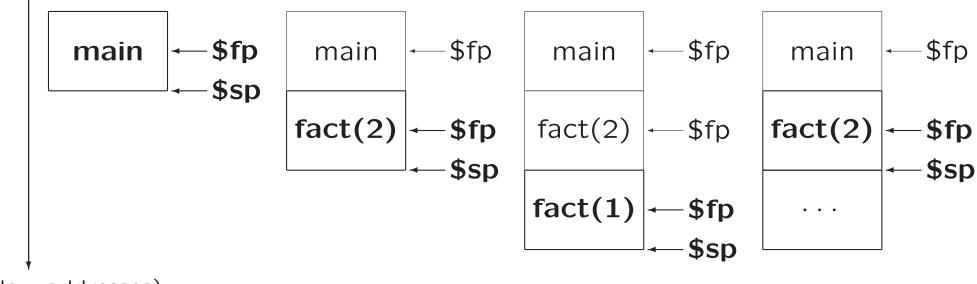
- The MIPS architecture has a separate coprocessor which deals with floating point arithmetic.
- Provides 32 new registes: **\$f0 \$f31**
- Single (32-bit) and double (64-bit) precision
- Examples:
  - li.s \$f2, 8.32
  - li.d \$f4, 3.14159265
  - add.d \$f6, \$f4, \$f4

# **Stack Frames**

(also known as "Activation records")

```
unsigned fact(unsigned x) {
  if (x == 1)
    return 1;
  else
    return fact(x-1)*x;
}
```

```
(high addresses)
```



(low addresses)