`CodeGen.scala`

Generating MIPS code given a AST

MIPS Registers

MIPS instruction uses 5 bits for register addressing, so there can be 2^5 = 32 registers

Number	Name	Use	Preserved across function calls?
0	`$zero`	constant 0	—
1	`$at`	assembler temporary	no
2, 3	`$v0, $v1`	function return values	no
4 - 7	`$a0 - $a3`	function arguments	no
8 - 15	`$t0 - $t7`	temporaries	no
16 - 23	`$s0 - $s7`	temporaries	yes
24 - 25	`$t8 - $t9`	temporaries	no

MIPS Review

Concerning `Cint_lit`

In the code generation phase, we need to dump the all the literals in `.data` section

Concerning `Cstring_lit`

The same here, for strings it similar layout, except that for non-printable characters we need to use their ASCII representation. Also, we should not forget 0 to tell assembler that string ended.

What does `.align` mean?

Concerning `.align`

The "word"s should be in a *word boundary* meaning in MIPS which is a 32 bit we need to have words as 4 bytes, next word as the next 4 bytes and etc.

`ClassameTable`

Table of class names used in the program

Dispatch table for `Symbol`

Object prototype for `Symbol` or attribute table

How to use class prototype to create the new object

Offsets

We use the combination of offsets and labels to use the static data (i.e. dispatch table, class table and etc.).

- Note that addresses are in bytes in MIPS