Abstract Robot Control Language (ARCL)

This is the intermediate language used for SCARL. This language is structed as to be suitable to be converted to either assembler or to code that is used for robotics control. The abstract machine is one that utilizes ports and has a simple memory layout. We define the instruction set for the Abstract Robot here and provide the instruction’s close relative within the Atmel AVR Instruction Set.

Important in this language is the flags for the SCARL linker that says to generate code for a specific device action. It is up to the SCARL linker (it will be a compiler-linker combination) to perform the final conversion when it runs into such a directive, as the implementation of the device may vary from machine to machine.

Description of the abstract machine that ARCL runs on:

A machine that:

1. Has a 8 distinct 16 bit registers.
2. Has a stack with unbounded memory.
3. Has a space of read-only memory that is set before execution of the program.
4. Each atomic object is a 2 byte word that can be interpretted in multiple different ways
5. Has a set of peripheral pins that may be set to input or output, and have power on, or read the power from it

Instruction Set Table

|  |  |
| --- | --- |
| **Expression Instructions** | **Action** |
| ADD | Pop the last two items on the stack, interpret them as numbers, add them, and then push the result onto the stack |
| SUB | Same as ADD, but treated as a subtraction |
| MULT | Same as ADD, but multiplies the values intead |
| DIV | Same as ADD, but performs integer division |

|  |  |
| --- | --- |
| **Memory Instructions** | **Action** |
| PUSHL *lit* | Pushes the literal *lit* onto the stack |
| POP *reg* | Pops the top of the stack and stores it into a valid register *reg* |
| PUSH *reg* | Pushes the value stored in register *reg* onto the stack |
| MOV *src dst* | Moves data from the *src* register to the *dst* register |
| LOADL *reg lit* | Loads the register *reg* with the literal *lit* |
| LOADF *reg offset* | Loads the value stored *offset* values away from the top of the stack and loads it into register *reg* |
| LOADR *reg offset* | Finds the value from static storage that is *offset* values away from the beginning of static storage and stores it into register *reg* |
| CHANGF *reg offset* | Goes to the value located *offset* values away from the top of the stack and changes it so that it is equal to the value located in the register *reg* |

|  |  |
| --- | --- |
| **Program Control Instructions** | **Action** |
| LABEL *label* | Defines a label at this spot and calls it *label* |
| PROC *label* | Begins a procedure definition and calls it *label* |
| RET | Returns control back to the callee from a procedure call |
| CALL *label* | Creates an activation record for the procedure called *label* and transfers program execution to that procedure |
| JMP *label* | Jumps to the label *label* |

|  |  |
| --- | --- |
| **Device Control Instructions** | **Action** |
|  |  |
|  |  |
|  |  |
|  |  |