ICFP Programming Contest 2014

Additional Specification

This specification is in addition to the original specification that was used in the lightning round.

Full Round Summary

For the full round we will stick to the "classic" version of the game where it is just a single Lambda-Man against the ghosts, but your challenge now is to write an AI for the Lambda-Man *and* AIs for the ghosts. Your Lambda-Man AI will play matches against ghost AIs from an opposing team.

You may submit up to 4 ghost AI programs.

Each match is made up of two encounters with another team:

- once with the ghosts running your ghost AI code and Lambda-Man running your opponent's Lambda-Man AI code, and
- once with the ghosts running your opponent's ghost AI code and Lambda-Man running your Lambda-Man AI code.

Each encounter in a match will use the same map supplied by the judges.

The Lambda-Man who scores the highest in a match wins against the other. Note that whether Lambda-Man completes the level or not does not matter, only the score is important.

The full round runs for the full 72 hours of the competition, so submissions must be in before 12:00 UTC 28/07/2014.

Tournament scoring

The overall winners for the full round are then determined using a tournament algorithm based on individual win/lose/draw encounters between teams. Later tournament rounds will use harder maps to help distinguish good teams. The score within games may be used for tie-breaking.

Suggest your own maps

We invite you to submit maps for us to consider. We may choose to use them in the latter stages of the tournament, or indeed to help score the lightning round.

The undocumented second argument

We have discovered what the undocumented second argument to the Lambda-Man initialisation function is. It turns out to be an encoding of the ghost programs!

The mind boggles at why they included this little feature. Our best guess is that it was part of an attempt by the Lambda-Man AI authors to get one over on their rivals in the ghost AI team. Perhaps out of work hours they competed by running the game with AI against AI with no human player.

So the second argument is a list containing the program for each ghost in the map. Note that there may be duplicates in this list because there can be more ghosts than ghost programs, in which case ghosts are assigned programs cyclically.

Each program is a list of instructions. Each instruction is encoded as follows:

- (opcode, args)

That is, a pair consisting of an integer code and a list of arguments.

The opcode is the enumeration from 0 of the instructions, in the order defined in the GHC spec.

Instruction	Opcode
MOV	0
INC	1
DEC	2
ADD	3
SUB	4
MUL	5
DIV	6
AND	7
OR	8
XOR	9
JLT	10
JEQ	11
JGT	12
INT	13
HLT	14

The list of arguments depends on the instruction. Each argument is encoded in one of the following forms

Instruction argument	Encoding
register	(0, register number as an integer)
register-indirect	(1, register number as an integer)
constant	(2, constant as an integer)
address	(3, address as an integer)
INT number	interrupt number as an integer
JLT/JEQ/JGT target address	address as an integer

The register number is 0 to 7 for registers A to H and 8 for the PC register.

For example, `HLT` is simply `CONS 14 0`, that is op code 14 with the empty list of arguments ([] being encoded as 0). While `MOV A [B]` is encoded as

```
CONS 0                        -- the MOV opcode
     (CONS (CONS 0 0)         -- first arg, reg A
           (CONS (CONS 1 1)   -- second arg, memory refrence [B]
                 0))          -- the end of the list []
```

Submission procedure

The submission procedure is the same as for the lightning round except that there are extra files to include in your submission.

- Submission form

Solution format

For the full round, your submission file (.zip or .tar.gz) should have the following format:

- Subdirectory `solution` that contains your solution files:
  - `lambdaman.gcc`

    - `ghost0.ghc`
    - `ghost1.ghc` (optional)
    - `ghost2.ghc` (optional)
    - `ghost3.ghc` (optional)

You can submit between 1 and 4 ghost programs. Please number them sequentially.

- An optional `maps` subdirectory that contains any map files you wish to suggest. These should all be valid map files with `.txt` file extensions.

- Subdirectory `code` with the source code you wrote to help you prepare your solution and any auxiliary material that can be helpful to the judges to build your code. You should include a `README` file here with any documentation/description of your solution that you wish to share with the judges.

If you submit maps you may also like to tell us what you like about your suggestion.

Site proudly generated by Hakyll using Haskell