The Evolution of Vmgen

M. Anton Ertl
TU Wien

Input

```
+ ( n1 n2 -- n ) core plus
  n = n1+n2;
```

Output: Execution

```
I_plus: /* + ( n1 n2 -- n ) */
{
Cell n1;
Cell n2;
Cell n;
n1 = (Cell) sp[1 ];
n2 = (Cell) TOS;
sp += 1 ;
fp += 0 ;
NAME("+")
{
n = n1+n2;
}
NEXT_P1;
TOS = (Cell)n;
}
NEXT1_P2;
```

Original Goals

- Ease Gforth implementation
- Reduce bugs
- Less maintenance overhead

New Goals

- Ease virtual machine implementation (generalization)
- Support static superinstructions (efficiency)

Allow user configuration

- Old: two stacks (data and FP)
  15 type prefixes
  hard-coded into generator
- New:

```
\E stack data-stack   sp Cell
\E stack fp-stack     fp Float
\E stack return-stack rp Cell
...
\E s" Cell" single data-stack type-prefix n
...
\E return-stack stack-prefix R:
```

Stack prefixes

- Old:

```
>r ( w -- )     core to_r
*--rp = w;
```

- New:

```
\E return-stack stack-prefix R:

>r ( w -- R:w ) core to_r
```

Instruction Stream

- Old:

```
lit (    -- w ) gforth
w = (Cell)*ip++;
```

- New:

```
\E inst-stream  stack-prefix #

lit ( #w -- w ) gforth
```

Superinstructions

`lit+ / lit_plus = lit +`

`5 +` compiles to $\boxed{\texttt{lit+}}\boxed{\texttt{5}}$ instead of $\boxed{\texttt{lit}}\boxed{\texttt{5}}\boxed{\texttt{+}}$

New output features

- generate VM code
- disassemble VM code
- trace VM code
- profile VM code

Usage

Old:

```
gforth prims2x.fs -e "c-flag on s\" prim.i\" out-filename 2!" \
 -e "s\" prim.vmg\" ’ output-c ’ output-c-combined process-file bye"
```

New:

```
vmgen prim.vmg
```

Conclusion

- Generalize Gforth’s primitive generator
- Allow user configuration
- More stacks, instruction stream
- Superinstructions
- Output features
- Usage