ePS32 Metacompiler

Silicon Valley Forth Interest Group

October 23, 2010

Dr. C. H. Ting

Extensibility of FORTH

- **Define colon words, variables, constants**
- **Define assembly code words**
- **Metacompile complete target systems**
- **Design FORTH CPU**

FORTH Metacompiler

- Generate a new FORTH system on an existing FORTH system
- Target system may or may not have the same CPU
- Target system may or may not have the same FORTH architecture
- Ideally suitable to develop embedded systems

ePP32 Metacompiler

- **eP32 Metacompiler has these components:**
  - **Metacompiler**
  - **Assembler**
  - **eP32 Kernel**
  - **eForth Interpreter**
  - **eForth Compiler**
  - **eP32 Simulator**

Metacompilation

- **FORTH uses commands , (comma) and C, (c-comma) to build new words in dictionary**
- **Metacompiler redefines , and C, to build new target words in an array which will become the dictionary in target system**

Target Image

```
CREATE ram  8000 ALLOT
: RAM@    ram +  @ ;
: RAM!    ram +  ! ;
```

ePs32 Assembler

- Assemble 1-5 instructions in a 32 bit program word
- eP32 has two types of instructions :

Long Instructions

00 cccccc aaaaaa aaaaaa aaaaaa aaaaaa

Short Instructions

00 cccccc cccccc cccccc cccccc cccccc

Short Instructions

```
: INST CONSTANT DOES> R> @ ,I ;

08 spread inst ldrp  09 spread inst ldxp ( 0A spread inst ldi) 0B spread inst ldx
 0C spread inst strp  0D spread inst stxp  0E spread inst rr8   0F spread inst stx
 10 spread inst com   11 spread inst shl   12 spread inst shr   13 spread inst mul
 14 spread inst xor   15 spread inst and   16 spread inst div   17 spread inst add
 18 spread inst popr  19 spread inst xt    1A spread inst pushs 1B spread inst over
 1C spread inst pushr 1D spread inst tx  ( 1E spread inst nop ) 1F spread inst pops
```

Long Instructions

```
: jump CONSTANT DOES> anew R>
   FFFFFF AND @ OR #, ;
      0 JUMP bra
2000000 JUMP bz
3000000 JUMP bc
4000000 JUMP call
5000000 JUMP next
```

Structured Assembler

```
: if      h @ 0 bz ;  ( 5F80000 )
: ifnc    h @ 0 bc ; ( 5F40000 )
: skip    h @ 0 bra ;        ( 5FC0000 )
: then    begin OVER ram@ OR SWAP ram! ;
: else    skip  SWAP then ;
: while   if SWAP ;
: whilenc ifnc  SWAP ;
: repeat  bra then ;
: again   bra ;
: aft ( a -- a' a" )
   DROP skip begin SWAP ;
```

eP32 Kernel

- FORTH has been always a Virtual Machine, which interprets and compiles FORTH words.
- Even as eP32 is a true FORTH engine, it still must be converted to a Virtual FORTH machine, with a kernel of FORTH words implemented in eP32 machine instructions.

Low/High Level Words

```
: CODE
    makeHead begin .head CONSTANT
    DOES> R> @  call ;
: ::
    makeHead begin .head CONSTANT
    DOES> R> @  call ;
```

In eP32 system, code words and colon words are the same, all making subroutine calls.

eForth Interpreter

- Terminal IO
- Common utility words
- Multplication and division
- Memory read/write
- Number to text conversion
- Printing words
- Text to number conversion
- Text input
- Error handling
- Text interpreter

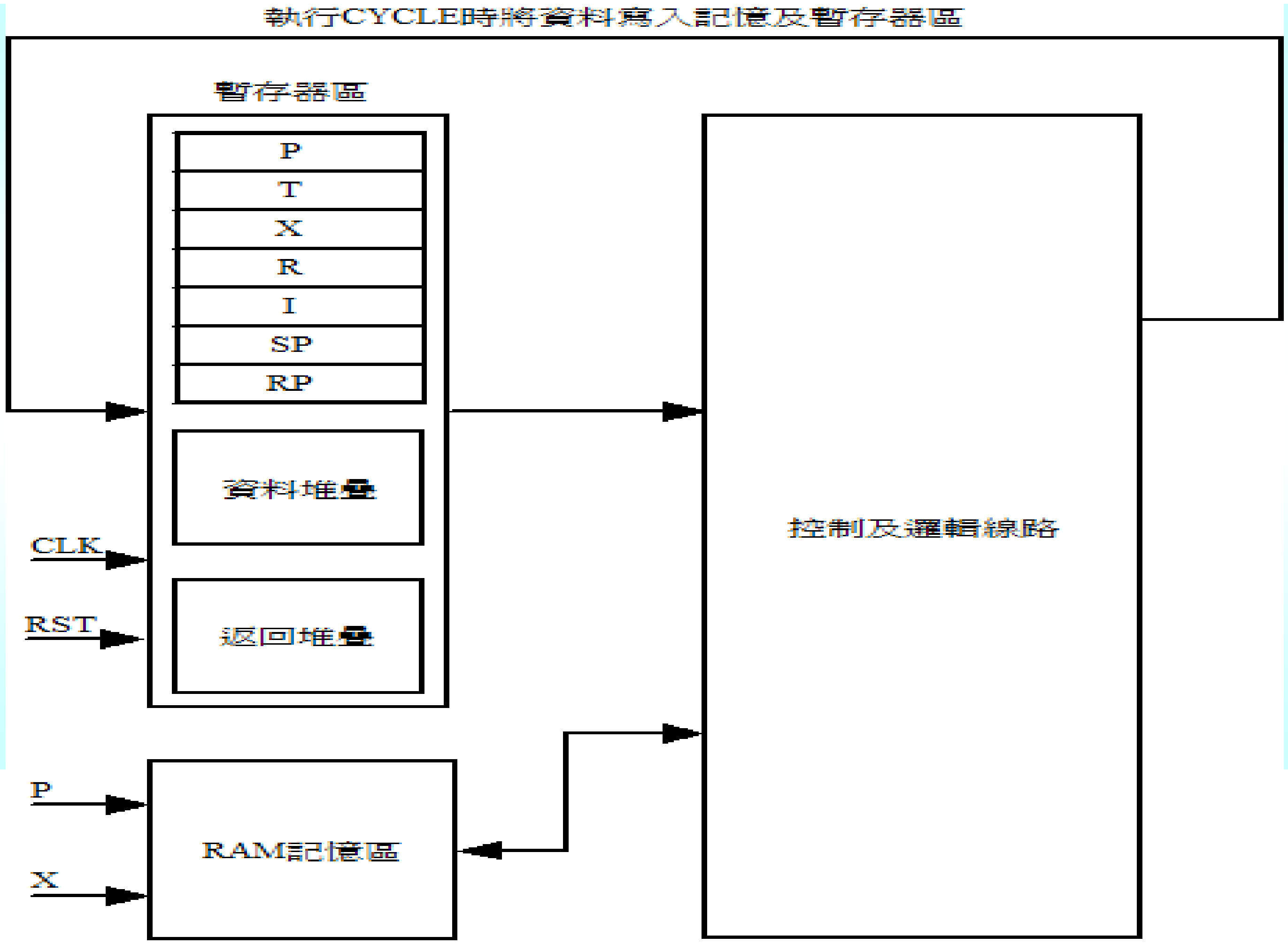
eForth Compiler

- Assembler
- Text compiler
- Tools
- Boot up
- Structural compiler
- Defining words
- eP32 Macros

eP32 Simulator

- Utilities
- Arrays and variables
- Finite state machine
- Machine instruction simulator
- User interface

eP32 Simulator



Simulation Commands

C:	Execute next cycle
S:	Display registers and stacks
D:	Display 8 program words
addr M:	Display memory
addr P:	Assign next program address
addr G:	Execute till address is G
RUN:	Single stepping

Demonstrations

- Metacompile a eP32 eForth system
- Examine symbol table
- Examine program memory
- Invoke simulator
- Compare simulator results and eP32 results

Questions?

Thank you very much!