Universidade do Minho

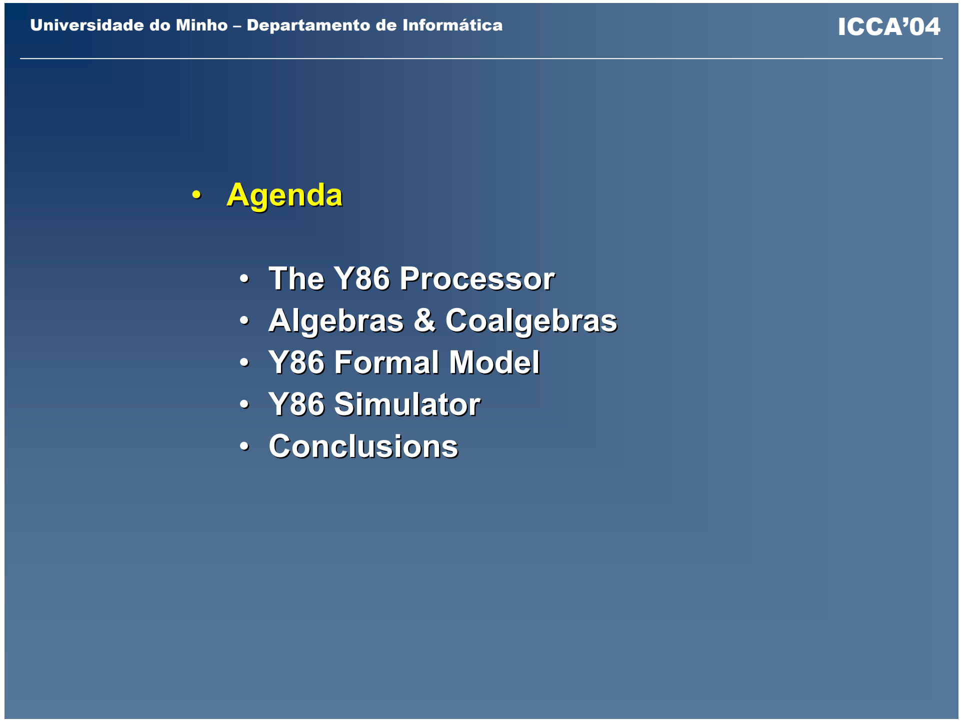
Departamento de Informática

A Coalgebraic Approach to the Y86 Processor Architecture

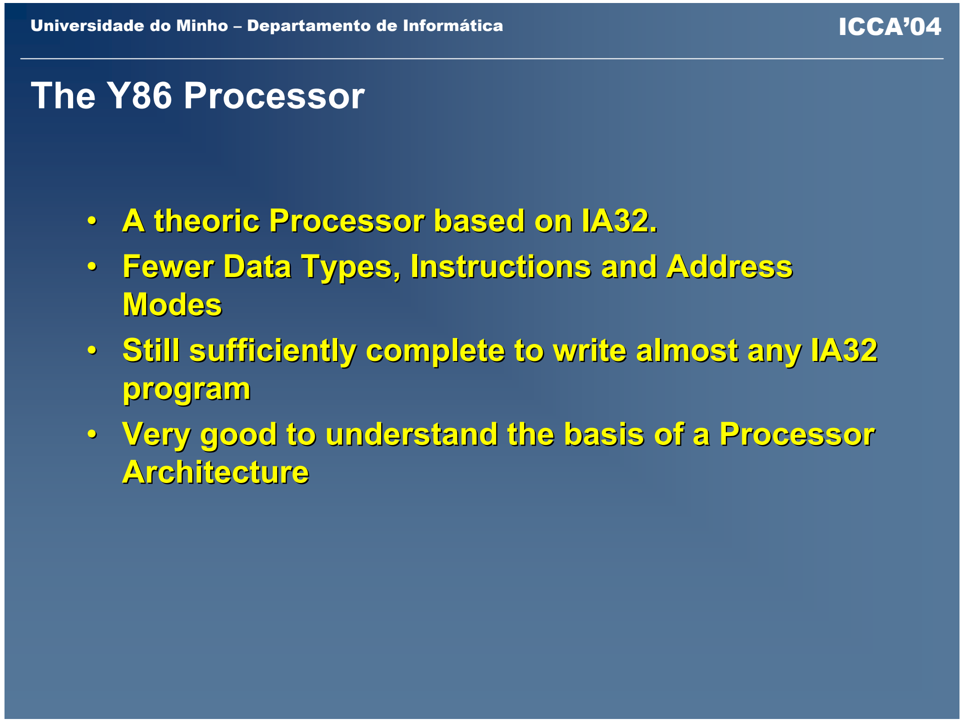
Nuno Rodrigues

5th Internal Conference on Computer Architecture (ICCA'04)

22 – Jan - 2004

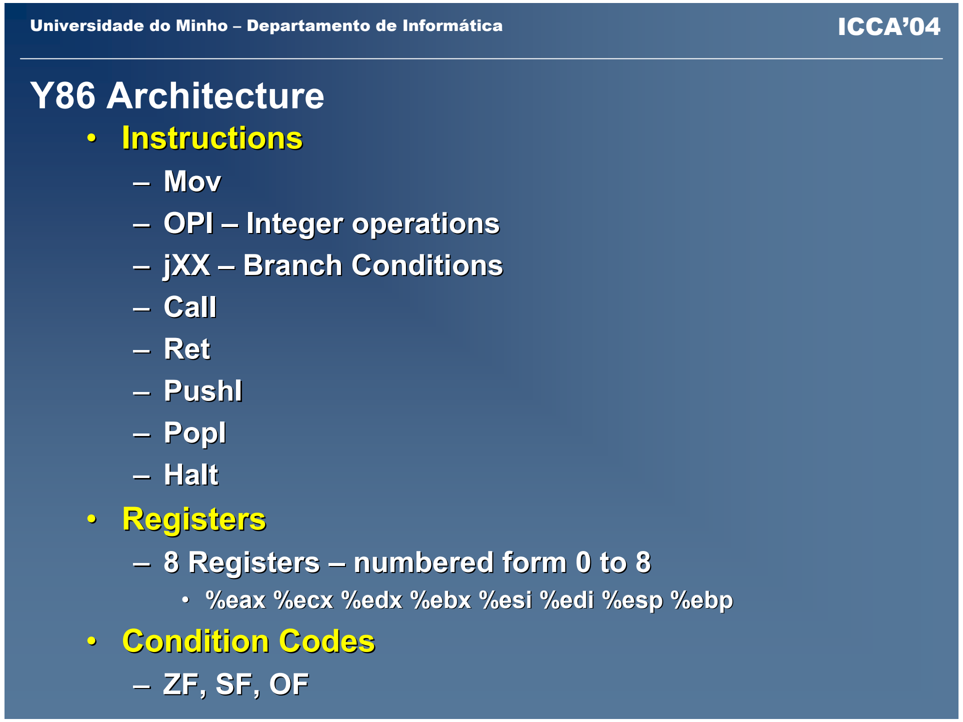


- Agenda
  - The Y86 Processor
  - Algebras & Coalgebras
  - Y86 Formal Model
  - Y86 Simulator
  - Conclusions



The Y86 Processor

- A theoric Processor based on IA32.
- Fewer Data Types, Instructions and Address Modes
- Still sufficiently complete to write almost any IA32 program
- Very good to understand the basis of a Processor Architecture



Y86 Architecture

- Instructions
  - – Mov
  - – OPl – Integer operations
  - – jXX – Branch Conditions
  - – Call
  - – Ret
  - – Pushl
  - – Popl
  - – Halt
- Registers
  - – 8 Registers – numbered form 0 to 8
    - %eax %ecx %edx %ebx %esi %edi %esp %ebp
- Condition Codes
  - – ZF, SF, OF

Algebras & Coalgebras and Functors

- Algebra
  - Sets with pre-determined rules
  - Main issue is to Construct new Elements to include in the Algebra

  F X = 1 + A * X * X

- Coalgebra
  - Main issue is to Destruct/Observe the Elements inside the Coalgebra
  - We are not concerned about the internal Set X

  F X = 1 + A * X * X

Y86 Formal Model

- Concepts to Formalize
  - Y86 Instructions
  - Y86 State
  - Y86 Program
  - Y86 Functional Simulator

Y86 Instruction

An Instruction Is Just a State modifier

$$\left.\begin{array}{l} \bullet\ S \longrightarrow S \\ \\ \bullet\ A \longrightarrow S^S \\ \\ \bullet\ A_1 * A_2 * \ldots. * A_n \longrightarrow S^S \end{array}\right\} \; F\ X = G\ X \longrightarrow G\ X$$

Haskell definition

```
data Y86Operation s v = Op (s -> (v, s))
```

Y86 State

- We just formalize what we really need to know about...

$$G\ X = A^8 = A * A * A * A * A * A * A * A$$

$$G\ X = A^8 * [F\ X]$$

Haskell definition

```
data Y86State = St { regs :: [Double],
                     mach :: [Y86Operation Y86State ()]
                   }
```

Y86 Program

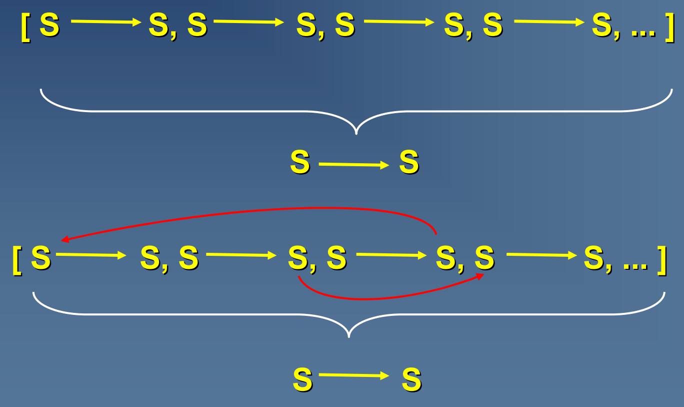
- A Y86 Program is a Sequence of Instructions
  - H X = 1 + F X * H X = [F X]

Haskell definition

```
type Y86Program = [ Y86Operation Y86State () ]
```

Y86 Functional Simulator

- Based on the Mathematical Concept of Monad


[S → S, S → S, S → S, S → S, ...]
S → S
[S → S, S → S, S → S, S → S, ...]
S → S

Haskell Simulator

```
instance Monad (Y86Operation s)
   where return a     = Op (\x -> (a, x))
         (Op f) >>= g = Op (\s -> let (a, s2) = f s
                                      Op fun  = g a
                                  in fun s2)


initSt = St (replicate 8 0)

prog2 :: [Y86Operation Y86State ()]
prog2 = [opIRMovl 1.0 1, opIRMovl 5.0 2, opIRMovl 8.0 3,
         opAddl 1 2, opJne 4 2 3]

exec2 = snd  (exec (sequence prog2) (initSt prog2))


  Result = [1, 8, 8, 0, 0, 0, 0, 0]
```

Conclusions and Future Work

- Formal Methods is a precise tool to capture a Processor architecture.
- Other Mathematic approaches could have been used.
- Base model to start reasoning about the Y86 Processor. Several simplifications took place.
- Bissimulation relation between Y86 programs
- Y86 Program calculus
- Introduce weight in the instructions arrows