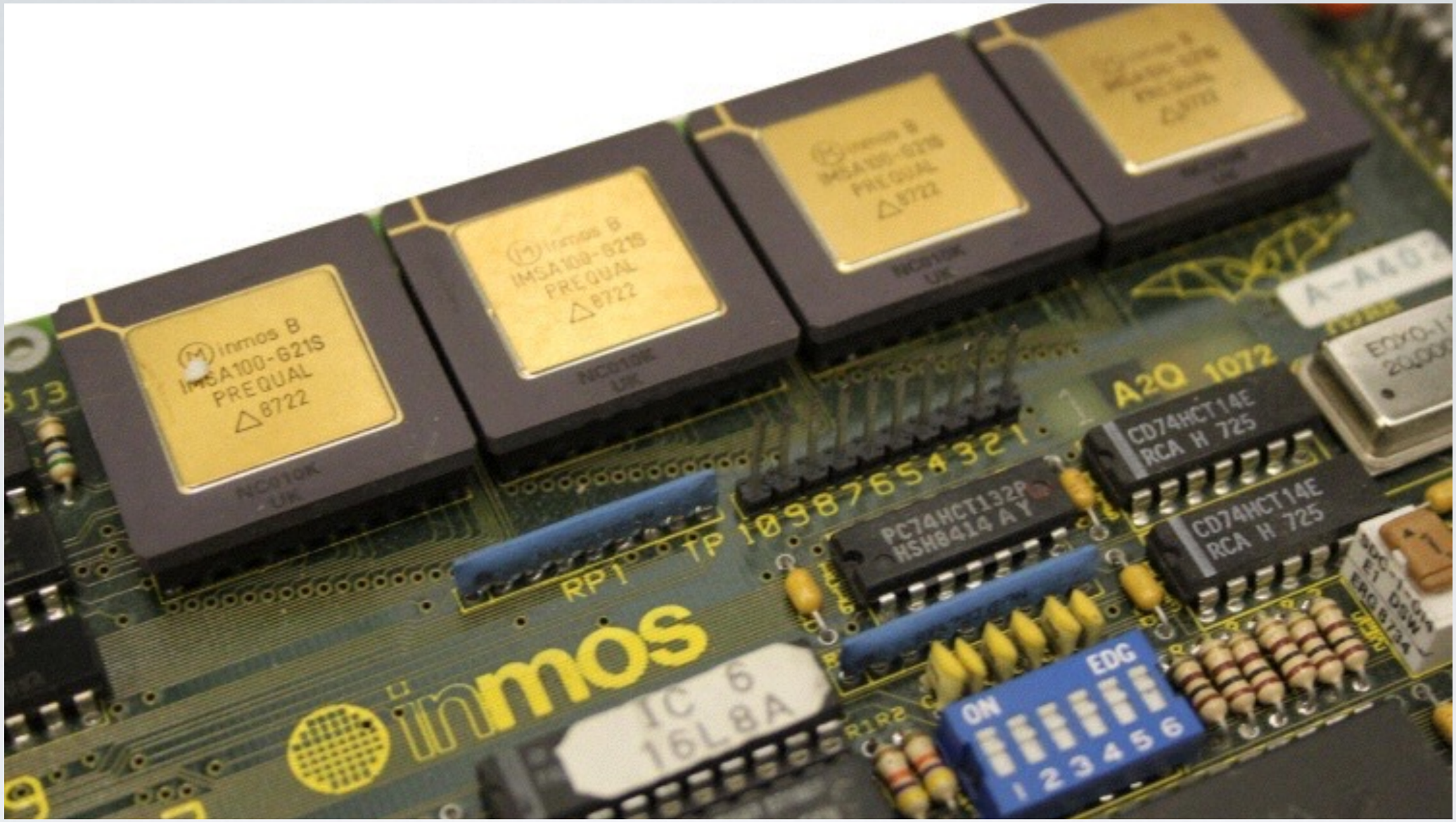


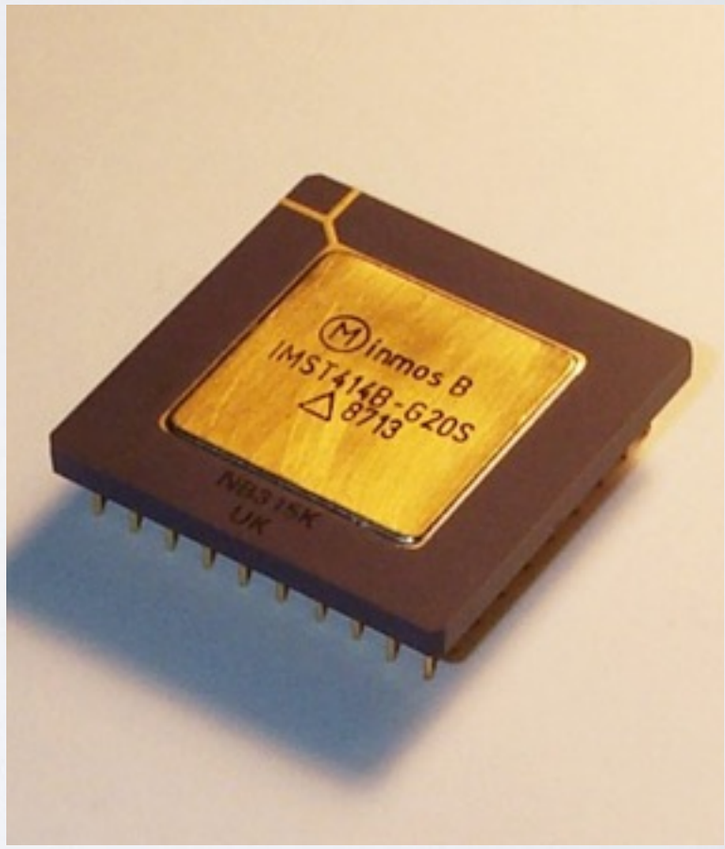
OCCAM

A software archaeology presentation

Andrei Cojocaru
Nicolae Pavel
MOC2

IN THE BEGINNING THERE WAS THE TRANSPUTER

- A general purpose microprocessor built specifically for parallel computing
- First released in 1984, shut down around 1989
- Was used for image processing, data acquisition, virtual reality, and even in space

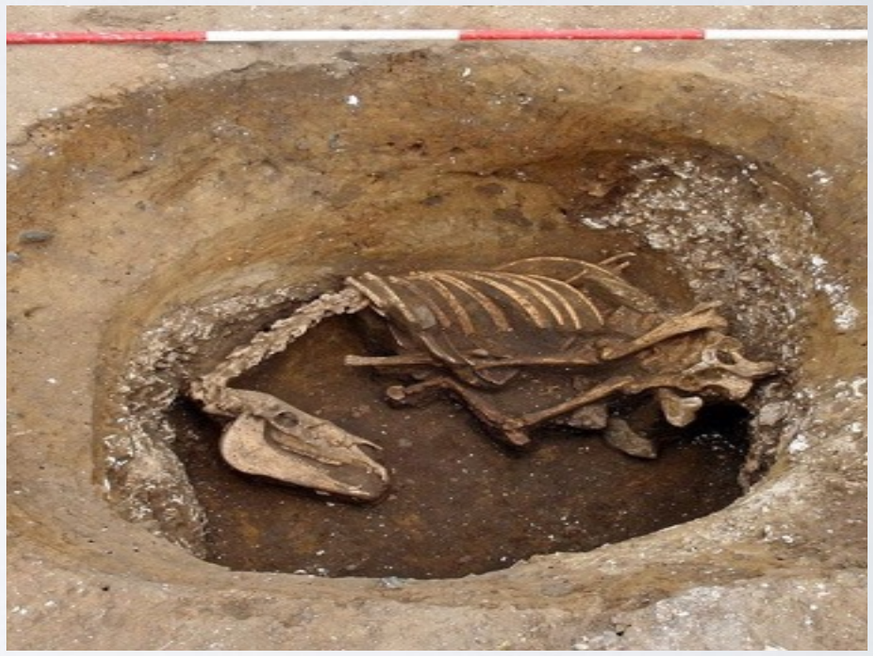


THE LEGACY: OCCAM

- An imperative programming language developed by INMOS to match their transputer's capabilities
- Built in parallelism and message passing between independent processes
- Pretty low level, comparable to early C

ARCHAEOLOGY: A WHINE

- One modern-ish version of Occam, the Kent Retargetable occam Compiler
- OS X build requires installing an old version of Apple's GCC, compiling an old version of LLVM with it, and only then compiling the occam compiler
- Linux build only works out of the box with an Ubuntu from 2011
- Windows… XP… is supported



HELLO WORLD

```
#INCLUDE "course.module"
PROC hello (CHAN BYTE out!)
  out.string ("Hello, world!*n", 0, out!)
:
```

- Keywords are CAPITALIZED
- Strings are byte arrays
- Statements are ***processes***
- out! is a communication ***channel*** between processes, carrying BYTEs
- the ! signifies sending, ? is receiving
- indentation is 2 spaces **exactly**

SEQ AND PAR

```
#INCLUDE "course.module"
PROC sequential (CHAN BYTE
 out!)
  INT x, y, z:
  SEQ
    x := 2
    y := 3
    z := 4
:
```

```
#INCLUDE "course.module"
PROC parallel (CHAN BYTE
 out!)
  INT x, y, z:
  PAR
    x := 2
    y := 3
    z := 4
:
```

```
#INCLUDE "course.module"
PROC parallel (CHAN BYTE
 out!)
  INT x, y, z:
  PAR
    x := 2
    x := 3
    z := 4
:
```

- Types: INT, BYTE, REAL32, REAL64
- Sequentiality is explicit with the SEQ ***constructor***
- Replacing it with PAR executes all ***processes*** in parallel
- Writing to the same variable in parallel is impossible, the right box is a compile error!

CHANNELS

```
#INCLUDE "course.module"
PROC sender (CHAN INT write!)
  INT seed:
  SEQ
    seed := 5000
    WHILE TRUE
      INT x:
      SEQ
        x, seed := random(256, seed)
        write ! x
:
```

```
PROC receiver (CHAN INT
 read?, CHAN BYTE out!)
  INT val:
  WHILE TRUE
    INT val:
    SEQ
      read ? val
      out.int(val, 0, out)
      out.string("*n", 0, out)
:
```

```
PROC mainisnotspecial
 (CHAN BYTE out!)
  CHAN INT comms:
  PAR
    sender(comms)
    receiver(comms, out)
:
```

- A ***channel*** is a pipe that allows one way communication between two processes
- sender writes random INT to a channel of… INT
- receiver reads INT from its read channel and writes their textual representation to the out channel
- There is no special name for the main procedure of a program, the last defined is executed first
- mainisnotspecial defines the channel that will pass data between sender and receiver, then runs both in parallel