# Progetto SO 2021/2022

December 2, 2021

# Panda+

- Il progetto del corso di Sistemi Operativi consiste nella creazione di un sistema operativo semplificato ma completo.
- Il progetto e' a fine principalmente didattico ma costruito per aderire il piu' possibile a un caso d'uso reale.
- Lo sviluppo verra' scandito da 3 livelli di specifiche: fase 1, fase 2 e fase 3.
- L'ambiente di riferimento per l'esecuzione del vostro software e' $\mu$MPS3.

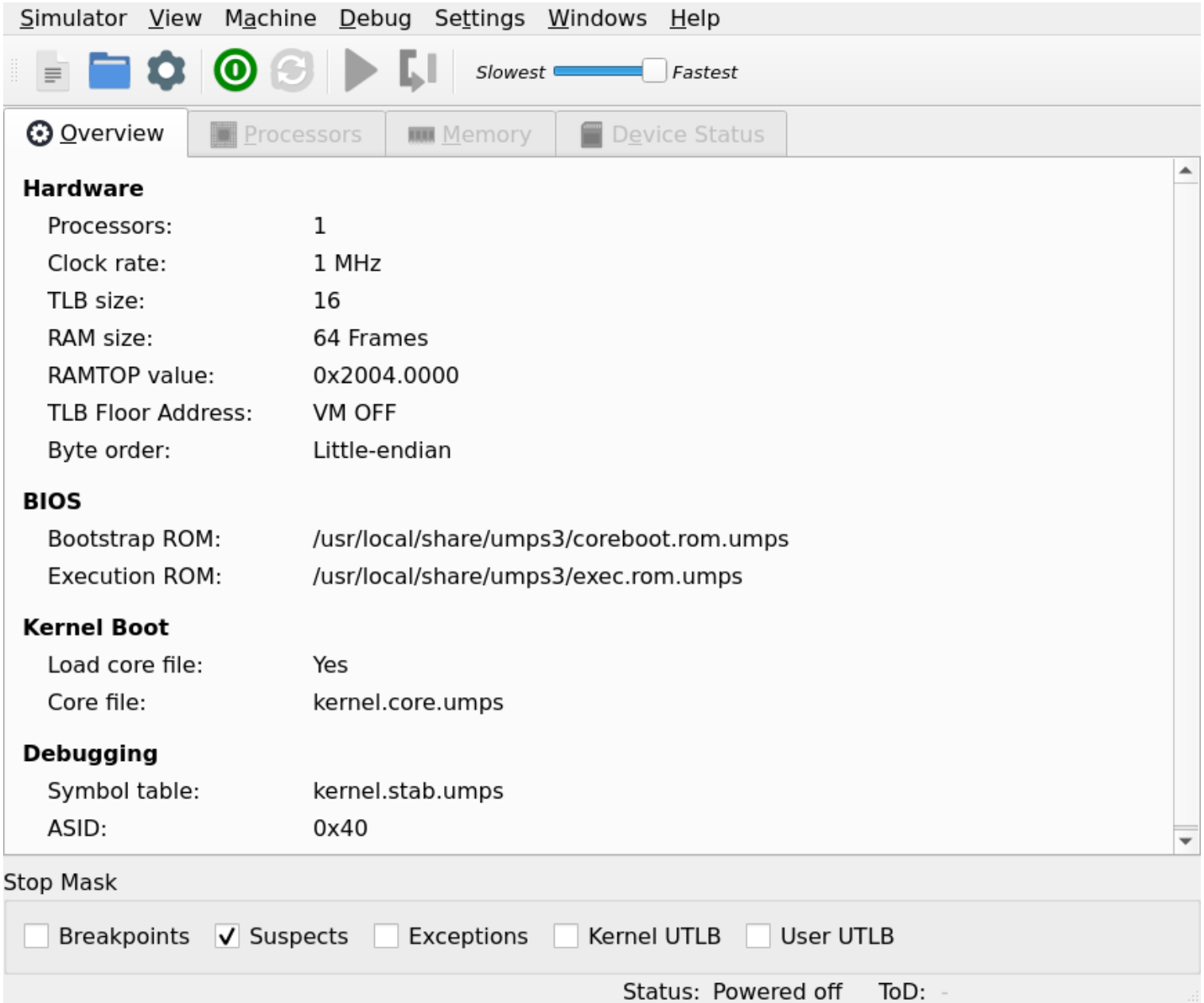
# Sviluppo di un Sistema Operativo

- Un approccio diverso dal solito: sviluppo firmware (quasi) bare metal
- Non c'e' nessun intermediario tra lo sviluppatore (voi) e l'hardware
- Anzi, il vostro compito e' proprio quello di costruire un intermediario (il SO)
- **Il vostro software e' l'unico attore che pilota la CPU**

# Bare Metal vs Userspace

- Accesso diretto alla memoria fisica (vs memoria virtuale)
- Interazione con i registri (vs System Call)
- Cross-Compilazione (vs compilazione nativa)
- Responsabilita' di inizializzazione (vs sistema pronto)
- Nessuna libreria immediatamente utilizzabile
- Senza rete!

# ”Hardware”


Simulator View Machine Debug Settings Windows Help
Slowest
Fastest
Overview
Processors
Memory
Device Status
Hardware
Processors: 1
Clock rate: 1 MHz
TLB size: 16
RAM size: 64 Frames
RAMTOP value: 0x2004.0000
TLB Floor Address: VM OFF
Byte order: Little-endian
BIOS
Bootstrap ROM: /usr/local/share/umps3/coreboot.rom.umps
Execution ROM: /usr/local/share/umps3/exec.rom.umps
Kernel Boot
Load core file: Yes
Core file: kernel.core.umps
Debugging
Symbol table: kernel.stab.umps
ASID: 0x40
Stop Mask
Breakpoints
Suspects
Exceptions
Kernel UTLB
User UTLB
Status: Powered off
ToD:

# $\mu$MPS3

- Si tratta di un emulatore per architettura MIPS con interfaccia grafica integrata per l'esecuzione, l'interazione e il debugging del software.
- Alcuni celebri esempi di dispositibi basati su MIPS: le console Playstation 2, Playstation Portable e Nintendo64; la sonda spaziale New Horizons; l'autopilota della Tesla Model S.

$\mu$MPS3 e' costruito in equilibrio tra fedelta' all'hardware vero e proprio e il fine didattico. E' molto simile a un vero processore MIPS ma con alcune notevoli differenze, soprattutto nel funzionamento della memoria virtuale.

I dispositivi che mette a disposizione si controllano tramite un'interfaccia di registri semplificata, e la parte piu' bassa dell'inizializzazione del sistema e' gia' fornita (BIOS).

Ciononostante, il binario che viene eseguito e' a tutti gli effetti valido per l'architettura MIPS.

# Installazione

- `https://github.com/virtualsquare/umps3`

Le istruzioni presenti nel Readme indicano diverse strade per installare $\mu$MPS3. Sono presenti delle vesioni pacchettizzate per le maggiori distribuzioni Linux, e in ogni caso e' possibile compilare l'emulatore dirattamente dai sorgenti.

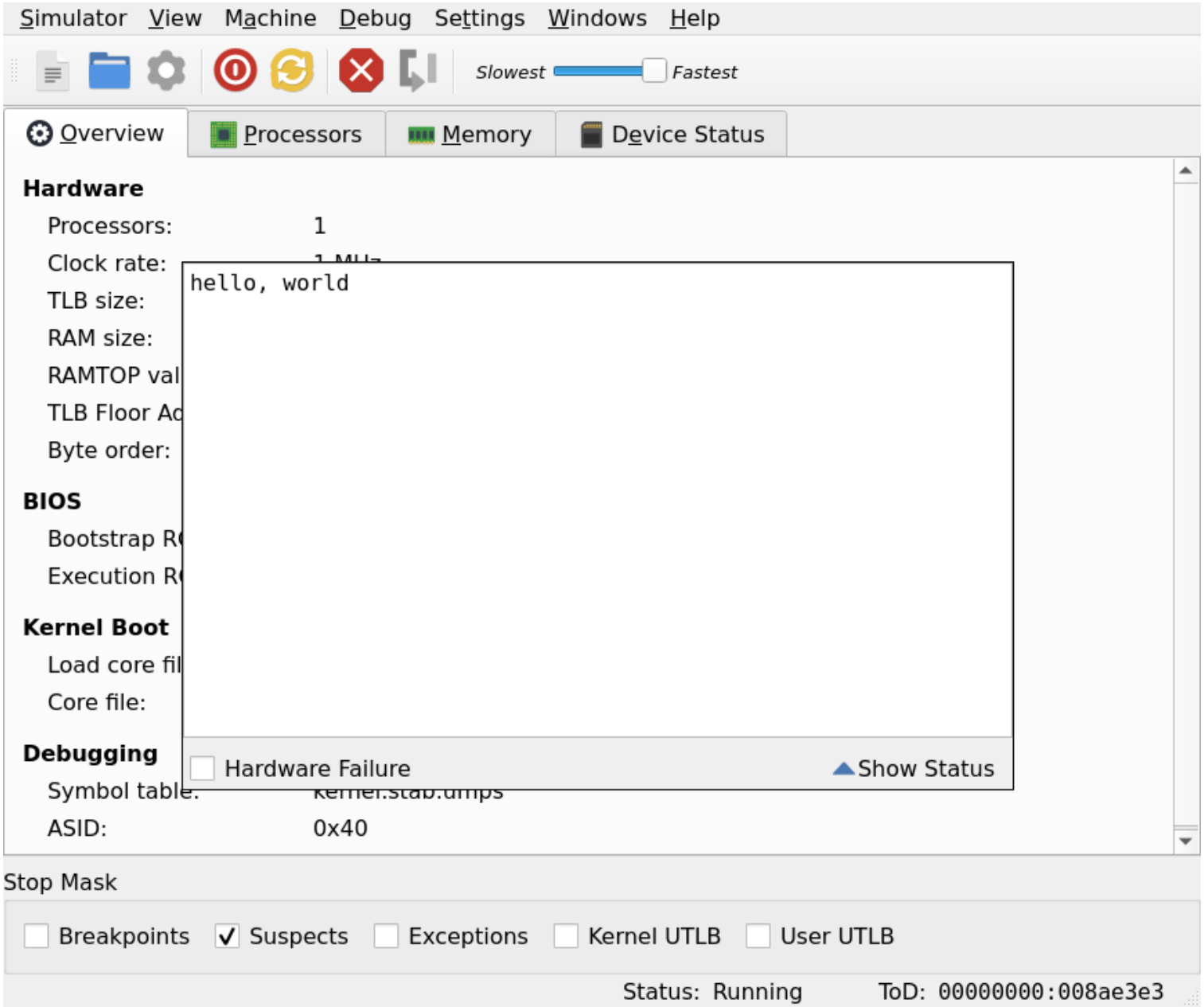
# Cross-Compiler

Per ottenere un eseguibile compatibile con $\mu$MPS3 e' necessario utilizzare un cross-compiler verso architettura MIPS.

Ne esistono diverse varianti, e in generale sono tutte adeguate. Sono reperibili nelle repository ufficiali di (quasi) tutte le maggiori distribuzioni di Linux.

Se cosi' non fosse, e' possibile compilare anche il compilatore dai sorgenti. E' un'occorrenza rara, ma puo' succedere (e.g. nel caso in cui voleste sviluppare il progetto su Raspberry Pi o simili).

# Hello world!


Simulator View Machine Debug Settings Windows Help
Slowest
Fastest
Overview
Processors
Memory
Device Status
Hardware
Processors: 1
Clock rate:
TLB size:
RAM size:
RAMTOP val
TLB Floor Ac
Byte order:
BIOS
Bootstrap R
Execution R
Kernel Boot
Load core fil
Core file:
Debugging
Symbol table:
ASID: 0x40
hello, world
Hardware Failure
Show Status
Stop Mask
Breakpoints
Suspects
Exceptions
Kernel UTLB
User UTLB
Status: Running
ToD: 00000000:008ae3e3

# Interazione con i device

$\mu$MPS3 espone 5 famiglie di dispositivi di input/output, ognuna delle quali e' popolata da 8 istanze separate. Ogni dispositivo e' manipolabile tramite 4 registri (con funzioni diverse a seconda della tipologia).

| **Field** | **Address** | **Name** |
|---|---|---|
| 0 | (base) + 0x0 | STATUS |
| 1 | (base) + 0x4 | COMMAND |
| 2 | (base) + 0x8 | DATA0 |
| 3 | (base) + 0xC | DATA1 |

# Operazioni specifiche dell'architettura

La manipolazione dei registri della CPU (normalmente) richiede l'utilizzo di istruzioni assembler. la libreria 'libumps.h' distribuita insieme a $\mu$MPS3 comprende funzioni invocabili da codice C per la lettura e la scrittura di tutti i registri necessari.

- getCAUSE()/setCAUSE()
- getSTATUS()/setSTATUS()
- STST()/LDST()
- HALT(), WAIT()

# Panda+

- Panda+: Evoluzione di Kaya O.S., a sua volta evoluzione di una lunga lista di S.O. proposti a scopo didattico (HOCA, TINA, ICARO, etc).
- Panda+ deve essere realizzato su architettura emulata
- Architettura basata su sei livelli di astrazione, sul modello del S.O. THE proposto da Dijkstra in un suo articolo del 1968 ...

# 6 Livelli di Astrazione

**Livello 6:** Shell interattiva

**Livello 5:** File System

**Livello 4:** Livello di Supporto

**Livello 3:** Kernel del S.O.

**Livello 2:** Gestione delle Code

**Livello 1:** Servizi offerti dalla ROM

**Livello 0:** Hardware

- Fondamenta note
- Fase 1

# 6 Livelli di Astrazione

**Livello 6:** Shell interattiva

**Livello 5:** File System

**Livello 4:** Livello di Supporto

**Livello 3:** Kernel del S.O.

**Livello 2:** Gestione delle Code

**Livello 1:** Servizi offerti dalla ROM

**Livello 0:** Hardware

- Fontamenta note
- Fase 2

# 6 Livelli di Astrazione

**Livello 6:** Shell interattiva

**Livello 5:** File System

**Livello 4:** Livello di Supporto

**Livello 3:** Kernel del S.O.

**Livello 2:** Gestione delle Code

**Livello 1:** Servizi offerti dalla ROM

**Livello 0:** Hardware

- Fontamenta note
- Fase 3

# Build System

- Siete fortemente incoraggiati a usare il build system che preferite
- Make e' uno standard di fatto, SCons e' piu' moderno e semplice da utilizzare, CMake e' piu' professionale...
- Riuscire a compilare correttamente il codice fa parte del progetto!

# (Il senso di un) Build System

- E' importante ricordare lo scopo e il senso di un Build System specializzato: minimizzare il numero di passaggi di compilazione a quelli strettamente necessari.
- Il progetto da sviluppare e' relativamente piccolo; anche nella sua fase finale e su una macchina poco performante la compilazione non richiedera' piu' di qualche secondo
- Ciononostante e' importante sfruttare appieno il Build System prescelto

# Risorse

- Il manuale di $\mu$MPS3
- Repository Github
- Virtualsquare
- Il sito del corso
- La guida di Pandos (precedente iterazione di Panda+, ancora rilevante)
- I progetti degli anni passati!

# Consegna

Prossimamente la presentazione di fase 1.