Sviluppo del supporto per chiamate di sistema e processi utente su architettura RISC-V

Tesi di Laurea in

Ingegneria Informatica

Candidato
Francesco Barcherini

Relatore
Prof. Giuseppe Lettieri

UNIVERSITÀ DI PISA

Introduzione e Problema

- L'obiettivo è migrare il nucleo didattico da x86-64 a RISC-V relativamente a processi utente, primitive di sistema e procedure di inizializzazione.
- Principali problemi:
  - Caricamento del programma utente
  - Assenza della libreria standard
  - Paginazione diversa da x86-64
  - Interruzioni gestite via software e tramite i registri CSR
  - Framework delle chiamate di sistema
  - Verifica del funzionamento del sistema
  - Organizzazione del codice

Soluzioni

- Moduli *sistema* e *utente* collegati separatamente
  - Modulo utente caricato dal modulo sistema
  - Costruttori del C++ eseguiti all'avvio
  - Creazione libreria `libce`
  - Definizione di spazio virtuale dei processi, spazio condiviso, heap e bugfix memoria virtuale
  - Aggiunta dei log sul terminale per testare il funzionamento del sistema
- Processi
  - Aggiunta al descrittore di `epc`, `satp` e `spie`
  - Implementazione salva/carica stato e creazione dei processi
- Chiamate di sistema
  - Interruzioni da livello utente e supervisore
  - Gestione software delle primitive tramite registro `a7` e istruzione `ecall`
  - Meccanismo di chiamata e implementazione di alcune primitive

Risultati

- Tramite i log sul terminale verifichiamo
  - l'inizializzazione del sistema:

```
INF     ?       Running in S-Mode
INF     ?       PCI initialized
INF     ?       VGA inizialized
INF     ?       PLIC Initialized
INF     ?       Numero di frame: 312 (M1) 32456 (M2)
INF     ?       Allocata tabella root
INF     ?       Creata finestra sulla memoria centrale:  [0000000000001000, 0000000088000000)
INF     ?       Attivata paginazione
INF     -       Nucleo di Calcolatori Elettronici - RISC-V
INF     -       Heap del modulo sistema: [0000000080038000, 0000000080138000)
INF     -       Suddivisione della memoria virtuale:
INF     -       - sis/cond [0000000000000000, 0000008000000000)
INF     -       - sis/priv [0000008000000000, 0000010000000000)
INF     -       - io /cond [0000010000000000, 0000018000000000)
INF     -       - usr/cond [ffff800000000000, ffffc00000000000)
INF     -       - usr/priv [ffffc00000000000, 0000000000000000)
INF     -       mappo il modulo utente:
INF     -         - segmento utente  read-only  mappato a [ffff800000000000, ffff800000003000)
INF     -         - segmento utente  read/write mappato a [ffff800000003000, ffff800000004000)
INF     -         - heap:                                 [ffff800000004000, ffff800000104000)
INF     -         - entry point: 0xffff800000001000
INF     -       Frame liberi: 32183 (M2)
INF     -       Creato il processo dummy (id = 0)
INF     -       Creato il processo main_sistema (id = 1)
INF     -       Cedo il controllo al processo main sistema...
```

Risultati

- Tramite i log sul terminale verifichiamo
  - il funzionamento dei processi sistema e utente:

```
INF     1       Creo il processo main utente
INF     1       proc=2 entry=ffff800000001000(0) prio=1023 liv=0
INF     1       Cedo il controllo al processo main utente...
INF     1       Processo 1 terminato
INF     2       Heap del modulo utente: 0000000000100000 [ffff8000000034d0, ffff8000001034d0)
DBG     2       >>>INIZIO<<<
DBG     2       Creo il processo main_body utente
INF     2       proc=3 entry=ffff80000000132e(1000) prio=2 liv=0
DBG     2       Cedo il controllo al processo main_body utente...
INF     2       Processo 2 terminato
DBG     3       Creo il processo conta1
INF     3       proc=4 entry=ffff8000000012b4(500) prio=1 liv=0
DBG     3       Aspetto che il processo conta1 conti fino a 500
DBG     4       Fine conta: count = 500
DBG     3       Creo il processo conta2
INF     3       proc=5 entry=ffff8000000012b4(500) prio=1 liv=0
DBG     3       Aspetto che il processo conta2 conti fino a 500
INF     4       Processo 4 terminato
DBG     5       Fine conta: count = 1000
DBG     3       Test completato con successo
DBG     3       >>>FINE<<<
INF     3       Processo 3 terminato
INF     5       Processo 5 terminato
INF     0       Shutdown
```