Programmazione di Sistema

A.A. 2024-2025

Questo documento inizia con una breve descrizione di regole e tempi per assegnazione e realizzazione dei progetti. Seguono le descrizioni dei progetti stessi, con sezioni distinte per le parti:

- OS Internals (S. Azimi)
- Application Programming (M. Rebaudengo)

Regole comuni ai progetti proposti dal Prof. Sarah Azimi e dal Prof. Maurizio Rebaudengo

Si riassumono brevemente le regole operative, comuni alle due parti del corso. Successivamente si descrivono dettagli tecnici dei progetti proposti

- I progetti sono opzionali e la loro valutazione si somma a quella degli esami scritti.

- Ogni studente può ottenere l'assegnazione di un solo progetto nell'ambito del suo percorso di studi (non è quindi possibile cambiare progetto oppure rifarlo dopo averne già **richiesto** uno). In linea di massima il progetto dovrebbe essere assegnato durante l'anno accademico di "nuova frequenza". Tuttavia, al fine di tenere conto del fatto che più studenti frequentano un corso e/o partecipano alle attività formative in anni successivi a quello di "nuova frequenza", uno studente può essere equiparato a nuovo frequentante se non ha mai sostenuto alcun appello (ritirato = sostenuto).

- I progetti sono svolti in gruppi di 2 o 3 o 4 persone (non si esclude, in casi particolari, il lavoro fatto da una sola persona, ma non si può garantire, in tali casi, una riduzione del livello di difficoltà e/o del lavoro richiesto)

- I progetti 2024-2025 andranno completati e consegnati entro la sessione di esame di Febbraio 2025. L'inizio del nuovo corso, nell'a.a. 2025-2026 cancellerà automaticamente tutti i progetti ancora in sospeso. Per la consegna, in fase di assegnazione, vi verrà fornito l'accesso ad un repository Github privato. L'organizzazione di tale repository è a totale discrezione del gruppo.

- I progetti saranno valutati a seguito di una breve presentazione (un colloquio online che includa anche la dimostrazione del prodotto sviluppato) dei candidati, durante una sessione di esame. Affinché sia valutato un progetto in una data sessione, questo deve essere condiviso con il docente di rifermento **per il proprio progetto prima dell'esame scritto (di uno dei due, nel caso della sessione estiva) di quella sessione**. Per la sola sessione Estiva 2024, le due date sono fissate in alternativa al 25 Giugno 2024 ed al 11 Luglio 2024.

- Il voto può essere diverso tra studenti dello stesso gruppo, in quanto verranno valutate, nel colloquio orale, le singole persone e i rispettivi contributi al lavoro. La valutazione potrà variare

tra -2.0 e +6.0. Un progetto assegnato ma non completato sarà valutato con punteggio -2.0 (per tutti gli studenti del gruppo di lavoro). **Una volta valutato, il voto del progetto non ha scadenza.**

Per l'assegnazione, entro **Lunedi 3 Giugno** ogni gruppo che desidera fare un progetto è invitato ad inviare all'indirizzo sarah.azimi@polito.it un'e-mail con la seguente riga (il numero di matricole dipende da quanti studenti compongono il gruppo):
<matricola1> <matricola2> <matricola3> <titolo progetto scelto>
<id GITHUB matricola 1>
<id GITHUB matricola 2>
<id GITHUB matricola 3>

Sezione 1 - Sezione di Sistemi Operativi

Progetto 1.1: Porting OS161 su Architettura ARM e Implementazione su Xilinx Zynq®-7000 SoC

Project Overview: Nel corso di OS Internal (OS161), abbiamo evidenziato il design di OS161 per l'architettura MIPS. Questo progetto mira a esplorare la fattibilità di adattare OS161 per eseguirsi su architettura ARM. Nello specifico, modificheremo il design di OS161 per renderlo compatibile con i processori ARM, aprendo possibilità per il suo utilizzo in sistemi basati su ARM.

Project Tasks:

- **Comprensione dell'Architettura ARM:** Inizia comprendendo approfonditamente l'architettura dei processori ARM. Questo include la comprensione del instruction set, del modello di memoria, dei registri e di altre caratteristiche specifiche dell'architettura.
- **Analisi del Design di OS161:** Analizza il design esistente e la base di codice di OS161 per identificare le dipendenze e le assunzioni specifiche di MIPS. Ciò include la comprensione di come OS161 interagisce con l'hardware, gestisce gli interrupt, gestisce la memoria e esegue altre funzioni essenziali del sistema operativo.
- **Modifica del Design di OS161:** Modifica il design di OS161 per renderlo indipendente dall'architettura e compatibile con i processori ARM.

  Implementazione su Xilinx Zynq®-7000 SoC: Utilizza il processore ARM Cortex™-A9 all'interno del Xilinx Zynq®-7000 SoC per ospitare il OS161 modificato, che combina la logica programmabile di un FPGA con un processore ARM Cortex™-A9 dual-core (sarà utilizzato solo il core ARM).

Progetto 1.2: Adapting a Didactic Operating System for ARM and Integration with Xilinx Zynq®-7000 SoC

Project Overview: L'obiettivo di questo progetto è selezionare e adattare un sistema operativo didattico adatto per i processori ARM, per poi implementarlo sul System-on-Chip (SoC) Xilinx Zynq®-7000. Il SoC integra la logica programmabile FPGA con un processore dual-core ARM Cortex™-A9. Dopo l'implementazione, il progetto prevede l'aggiunta di una funzione mancante al sistema operativo scelto e l'integrazione in un'implementazione basata su ARM.

Project Tasks:

- Identificare un sistema operativo didattico adatto per l'adattamento all'architettura ARM.
- Utilizzare il processore ARM Cortex™-A9 all'interno del SoC Xilinx Zynq®-7000 per ospitare il sistema operativo selezionato.
- Scegliere una funzione mancante dal sistema operativo selezionato (puoi scegliere una delle funzioni studiate durante i laboratori) e implementare la funzionalità mancante e integrarla nell'implementazione basata su ARM.

Progetto 1.3: Implementare Nuove Features per la Reliability di Task sul Sistema Operativo FreeRTOS e Integrazione del Sistema Xilinx Zynq®-7000 SoC

Overview: L'obiettivo del progetto è lo sviluppo e l'integrazione di codice C per ambiente FreeRTOS. Si richiede di triggerare e catturare exception compatibili con il sistema operativo generate da una task in un possibile contesto di corruzione della memoria per eventi esterni. Si agirà quindi sullo scheduling di questa

modificandone la priorità, generando starvation. L'idea è quella di permettere al sistema il recovery e/o correzione della task. Non è richiesto quest'ultimo punto. Essenziale è l'integrazione del sistema FreeRTOS su chip ARM utilizzando la board Xilinx Zynq®-7000 SoC (che vi sarà eventualmente fornita).

Requisiti:

- Integrazione del sistema operativo FreeRTOS su board Xilinx Zynq®-7000 SoC.
- Capire come triggerare le exception del S.O.
- Sviluppare una funzione che permetta la modifica della priorità se un'exception è stata identificata, generando starvation.

****

Nota 1: Per ogni studente interessato a questi progetti, è disponibile un sistema embedded Zynq 7020 SoC integrato in una scheda di sviluppo PYNQ-Z2PYNQ-Z2, fornita dall'azienda Xilinx. La scheda Pynq incorpora un microprocessore ARM che può essere utilizzato per far eseguire il sistema operativo in studio.

Nota 2: La board utilizzata ha una community molto attiva sulla rete. Diversi tutorial su l'ambiente di sviluppo e l'implementazione di funzioni base sono disponibili senza difficoltà di ricerca. Di seguito sono linkati esempi di tutorial che potete utilizzare:

- Hello world: https://www.youtube.com/watch?v=Mb-cStd4Tqs
- FreeRTOS: https://www.youtube.com/watch?v=Zp_eR_DTRuQ
- Vitis Environment Overview: https://www.youtube.com/watch?v=LU9hP7KLDgE

Sezione 2 - Sezione di Programmazione

Progetto 2.1: Back-up di emergenza

Realizzare un'applicazione in Linguaggio RUST per PC utile per permettere di effettuare un back-up nel caso in cui lo schermo non sia agibile. Qualora l'utente voglia effettuare un backup su un disco esterno (chiavetta USB), dovrà utilizzare un comando convenzionale attraverso il mouse. La scelta del tipo di comanda è lasciata libera, ma potrebbe essere una X che comprende l'area dello schermo oppure la composizione di un rettangolo lungo i bordi dello schermo. Dopo aver inviato il comando di attivazione di backup, l'utente deve dare conferma attraverso un secondo comando tramite mouse (potrebbe essere corrispondente al tracciamento di un + o di un -). La sorgente del back-up può essere indicata in fase di definizione del tool (ad esempio in un file di configurazione). Si richiede anche che dopo che il comando di attivazione del back-up è stato riconosciuto, venga visualizzata nello schermo una finestra di conferma. Il programma potrebbe prevedere anche diverse tipologie di backup (ad esempio il contenuto di una cartella oppure solo i file di un determinato tipo).
Si richiede che tale applicazione sia installata in fase di bootstrap del PC e sia attiva in background. L'applicazione deve avere il minor consumo di CPU possibile. Per valutare il consumo di CPU si richiede che ogni 2 minuti sia salvato su un file di log il consumo di CPU effettuato dall'applicazione. Infine si vuole che al termine dell'esecuzione di backup, nella medesima chiavetta, sia creato un file di log che specifica la dimensione complessiva dei file salvati ed il tempo di CPU impiegato per il completamento delle operazioni di backup.

Progetto 2.2: Sistema di Fault-Injection per Applicazione Ridondata

Si chiede di realizzare un ambiente di fault injection per la valutazione della tolleranza ai guasti di un'applicazione che è stata irrobustita con la duplicazione automatica delle variabili.
Il progetto mira a raggiungere 2 obiettivi:

1. Modificare il codice sorgente attraverso la duplicazione di tutte le variabili ed il controllo della loro congruenza dopo ogni operazione di lettura, sfruttando il polimorfismo disponibile con il linguaggio RUST
2. Realizzare un ambiente di Fault Injection costituito da:
    a. Un programma ridondato come descritto precedentemente
    b. Una lista di guasti caratterizzati dal nome della variabile, dal tempo di iniezione e dal bit modificato
    c. Un Iniettore che scandisce la lista di guasti e per ogni guasto lancia il programma a cui esso verrà iniettato
    d. Un Analizzatore che raccogliere i dati sulla base del risultato dell'elaborazione.