Fase 1.5

Nuovo Makefile

Il Makefile delle precedenti fasi è stato modificato, rendendo ora possibile la compilazione senza dover specificare i nomi dei file, è quindi necessario posizionare i file sorgenti nelle rispettive directory src e include, in questo modo potranno essere riconosciuti e compilati.

A compilazione ultimata sarà disponibile una nuova directory chiamata build, la quale conterrà tutti i file frutto della compilazione (file oggetto e kernel per l'architettura scelta).

Come nelle precedenti fasi è possibile compilare grazie al comando make con due opzioni, alle quali se ne aggiunge una terza. Le opzioni ora disponibili, diversificate dal cross-compiler utilizzato sono quindi: - uarm: compilazione per architettura uARM tramite cross-compiler arm-none-eabi - umps: compilazione per architettura uMPS tramite cross-compiler mipsel-linux-gnu - umps1: compilazione per architettura uMPS tramite cross-compiler mipsisa64r6el-linux-gnuabi64

L'aggiunta della terza opzione è dovuta alla mancanza del pacchetto mipsel-linux-gnu in alcune distribuzioni Linux.

Difficoltà riscontrate

Durante lo svolgimento di questa fase ci siamo imbattuti in qualche difficoltà di seguito esposta:

Gestione degli interrupt del terminale:

La fase 1.5 richiede di attivare gli interrupt negli state_t dei 3 PCB e sottolinea che sarà sufficente gestirne solo uno, ovvero quello del timer. Ci siamo però accorti che avendo (tutti) gli interrupt abilitati, ogni volta che si compie un operazione sul terminale esso solleva un interrupt che dovrà poi essere gestito dal rispettivo handler. Non essendo richiesta nella fase 1.5 la scrittura di un gestore degli interrupt che non fosse quello del timer abbiamo allora pensato che il problema potesse essere risolto mascherando gli interrupt provenienti dai devices e lasciando attivo solo quello del timer, ma sarebbe venuta a meno la richiesta di avere tutti gli interrupt abilitati. La seconda soluzione sarebbe stata quella di scrivere un gestore degli interrupt del terminale, e pensando a quale fosse l'alternativa più adetta la nostra scelta è ricaduta sulla seconda opzione e abbiamo scritto anche un gestore degli interrupt del terminale. Su uARM non c'è stato nessun problema a far ciò. Quando viene chiamata "tprint", la funzione di libreria per stampare su uARM, un interrupt del terminale è sollevato e grazie alla funzione "terminalHandler" che si occupa di mandare l'acknowlegment quando necessario l'interrupt viene correttamente gestito. Il problema si presenta invece su uMPS perchè abbiamo verificato che a causa della sintassi di "termprint", la funzione NON di libreria per stampare su uMPS interferiva con una corretta gestione degli interrupt. Ciò perchè per un corretto funzionamento "termprint" utilizza il ritardo causato dallo stato busy del terminale. Inserendo la gestione dell'interrupt causato dall scrittura durante l'esecuzione di "termprint", i ritardi utilizzati per il corretto funzionamento vengono a mancare così interrompendo la stampa. Il risultato è un test correttamente eseguito ma che stampa solamente il primo carattere di ogni stringa. Per limitare il problema abbiamo allora pensato di mascherare l'interrupt del terminale su uMPS, venendo a mancare alla richiesta della consegna di avere tutti gli interrupt abilitati. In

questa maniera gli interrupt causati dal terminale su uMPS sono gestiti dal BIOS e ciò non interferisce con "termprint". Su uARM sono invece tutti correttamente abilitati.

Sovrapposizione delle stampe:

Su uARM i processi sovrappongono le prime 3 stampe perchè impiegano più del TIME_SLICE di 3 ms per essere completate, le successive vengono invece alternate correttamente. Aumentando il TIME_SLICE di qualche ms questo problema può essere ovviato.