Strutturazione del codice e compilazione

Per la consegna di questa fase si è deciso di suddividere i sorgenti del progetto in cartelle separate per una migliore organizzazione e comprensione.
I sorgenti veri e propri si troveranno all'interno della cartella `src`, gli header file nella cartella `include` e la documentazione nella cartella `doc`.
Il **Makefile** inoltre durante la fase di making creerà due sottocartelle all'interno di `src`: `obj` per i file oggetto e `kernel` per i file relativi al kernel.

Inizializzazione

Per la fase di inizializzazione dei processi abbiamo lasciato la possibilità di scegliere tra due **status**, il primo con **PLT abilitato** (come specificato nelle slides) ed il secondo con **PLT disabilitato**.
La conseguenza della scelta porterà all'uso di un **timer** rispetto ad un altro per l'assegnazione del **time slice** dei processi.
Nel secondo caso, infatti, verrà utilizzato il predefinito **interval timer** al posto del **process local timer**.
Lo status dei processi ha tutti i bit dell'**interrupt mask** abilitati tranne l'ottavo, quello del `terminal`, in quanto sarebbe altrimenti stato necessario gestire gli interrupt generati alla trasmissione di caratteri sul terminale.
In questa fase abbiamo deciso di fare uso di un **contatore di processi**: un intero che viene incrementato ad ogni istanziazione di un PCB che ci tornerà utile nella fase di scheduling.

Scheduling

Lo scheduler implementato è una semplice funzione che si occupa di controllare che ci siano processi istanziati o in stato `ready`.
Nel caso non ci siano più processi istanziati l'intero sistema si fermerà.
In caso contrario, prima di passare il controllo ad un nuovo PCB, lo scheduler sceglierà di utilizzare un timer invece che l'altro in base al bit `TE`, come spiegato in precedenza.
Questa scelta progettuale è puramente a scopo dimostrativo della corretta implementazione e gestione di entrambi i tipi di timer forniti da `µMPS2`.

Interrupts handling

La gestione degli interrupt è stata implementata sottoforma di **switch-case** per facilitare le successive fasi di consegna del progetto.
Attualmente tutti gli interrupt non gestiti porteranno ad una interruzione del sistema.
I due unici interrupt gestiti, quelli dei timer, sono uguali in quanto uguale è il loro scopo; l'unica differenza sta nel metodo con cui si segnala l' `ACK`.

Syscall handling

La gestione delle systemcall (e dei breakpoint) è implementata analogamente a quella degli interrupt per lo stesso motivo.
È presente un controllo sul tipo di eccezione qui sollevata, e nel caso sia una `SYSCALL` un ulteriore controllo si

assicurerà che non venga invocata senza averne i privilegi.
La syscall `TERMINATEPROCESS`, l'unica da implementare in questa fase, non è stata ideata per prendere in input un parametro (il PCB da terminare), ma solo per **auto terminare** il processo che la invoca.
Data questa premessa, per poter terminare anche la progenie, è stata implementata una versione ricorsiva della system call che fa uso del processo corrente, ovvero l'unico che può averla invocata in questa implementazione a monoprocessore.

Program Traps e TLB Management handling

Le ultime due eccezioni, non necessarie in questa fase, se invocate non fanno altro che segnalare il tipo di errore sul terminale e mandare il sistema in `PANIC`.
Questo per evitare che l'esecuzione porti ad uno stato incontrollabile ed impossibile da prevedere a priori.