Strutturazione del codice e compilazione

La strutturazione del codice per la consegna di questa fase è stata mantenuta uguale a quella della fase precedente.
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

L'inizializzazione dei processi rimane pressochè invariata, a differenza del fatto che questa volta anche l'ottavo bit dell'*interrupt mask* viene abilitato. Prima di mandare in esecuzione il processo principale *test*, viene avviato il timer del `System Clock`, che scatta ogni 100 ms. Quest'ultimo servirà affinchè la sesta Syscall venga effettuata correttamente.

Scheduling

Lo scheduler implementato nella fase precedente è rimasto pressochè invariato: una semplice funzione che si occupa di controllare che ci siano processi istanziati o in stato `ready`.
È stato necessario aggiungere la gestione dell'aggiornamento del tempo che i processi passano in **kernel/user mode**.

Interrupts handling

La gestione degli interrupt implementata sottoforma di **switch-case** nella fase precedente ha facilitato il lavoro per questa consegna.
L'unico interrupt non gestito è quello sollevato sulla linea `0`, in quanto viene usato in presenza di più processori e non riguarda quindi la nostra implementazione. Tutti gli altri interrupt vengono gestiti tramite l'apposita funzione:

- l'handler del **Process Local Timer** non ha subito modifiche implementative significative;
- gestendo l'**Interval Timer** verranno inoltre sbloccati tutti i processi bloccati sul semaforo del clock. Prima di far proseguire l'esecuzione, viene resettato il timer in modo che scatti dopo 100ms;
- l'handling dei successivi quattro device (*disk, tape, network, printer*) è identico: una volta trovati quali degli 8 device per linea hanno un `interrupt pending`, viene effettuato l'ACK;
- l'handler dei terminali è implementato in modo analogo ai precedenti, ma in questo caso viene controllato lo stesso device due volte (in ricezione ed in trasmissione).

Syscall handling

Anche la gestione delle systemcall è stata ereditata dalla fase precedente: per ognuna delle 10 Syscall viene invocata la funzione specifica. Nel caso di Syscall non riconosciuta o breakpoint viene passata la gestione all'handler di livello superiore nel caso sia stato specificato. In caso contrario il processo termina. Informazioni significative sull'implementazione delle System Call:

- per la corretta implementazione della SYS1 è stato necessario dover aggiungere tre campi alla struttura dei `pcb` . In ogni punto del kernel in cui avviene il passaggio da `user mode` a `kernel mode` o viceversa è stato necessarrio aggiornare questi tre campi;
- la SYS3 ha richiesto una notevole modifica del codice data la modifica delle specifiche. Questa implementazione è stata una tra le più difficoltose a causa di un errore nel codice che ci siamo portati dietro fin dalla prima fase. Dopo una lunga ricerca del problema siamo riusciti ad implementare il tutto correttamente.
- la SYS7 è stata impegnativa a causa del problema di dover distinguere tra device normali e terminali.

Program Traps e TLB Management handling

Gli ultimi due tipi di eccezione vengono gestiti in maniera simile: entrambi comportano la terminazione del processo o la delega della gestione ad un handler di livello superiore se specificato.