Da fare:

memoria virtuale

* Algoritmo di paging
* Il controller della memoria di uARM è dotato di un Translation Lookaside Buffer (sec 2.2.3) per rendere più efficiente la risoluzione degli indirizzi (dobbiamo usarlo), inoltre il coprocessore permette di adottare meccanismi di swapping (caricamento di memoria in memoria virtuale, penso) con algoritmi arbitrari
* Richiesto di far funzionare la memoria virtuale, con un algoritmo di paging first in-first out

Gestione dischi

* Usati come memoria ausiliaria per il meccanismo di swapping (sec 4.3.1)(in pratica non ho capito cosa effettivamente venga salvato, se le tabelle di paging, tutta le memoria o cosa) (un disco riservato a questi scopi)
* Usati come dispositivo di storage e scambio dati per i processi utente

Caricamento ed esecuzione dinamici

* Permette di caricare ed eseguire programmi caricati a run time da processi esterni (sec 2.2.2)
* Si possono avviare anche più programmi in parallelo in questo modo

Temporizzazione

* Implementare sleep(xxxx) con numero arbitrario di secondi (bisognerà fare una syscall per questo)
* Deve essere implementato con un area di memoria in cui vengono salvati i processi in attesa e un demone che periodicamente controlla se si devono svegliare

System call a livello utente

* int SYSCALL (READTERMINAL, char \*virtAddr) (SYS11): fa attendere il processo finche non legge un input da terminale (associato al processo)
* int SYSCALL (WRITETERMINAL, char \*virtAddr, int len)(SYS12): scrive su terminale (associate al processo) una stringa
* void SYSCALL (VSEMVIRT, int \*semaddr, int weight)(SYS13): operazione V dei semafori pesata (anzichè aumentare di 1, aumenta di ‘weight’) (I puntatori a semafori sono relativi alla memoria virtuale)
* void SYSCALL (PSEMVIRT, int \*semaddr, int weight): uguale ma con la P
* void SYSCALL (DELAY, int secCnt): fa sleep(secCnt)
* int SYSCALL (DISK\_PUT, int \*blockAddr, int diskNo, int sectNo): inserisce blocco nel disco
* int SYSCALL (DISK\_GET, int \*blockAddr, int diskNo, int sectNo): prende blocco da disco
* int SYSCALL (WRITEPRINTER, char \*virtAddr, int len): stampa len caratteri a partire da virtAddr, sul printer associato al processo (lo stato di successo/insuccesso è comunicato comme valore di ritorno)
* unsigned int SYSCALL (GETTOD): chiede di returnare il time of the day
* void SYSCALL (TERMINATE): fa terminare il processo corrente, se tutti i processi terminano, il sistema deve fare “shut down”

nota: gli input/ output delle syscall vengono messi nei registri a1,…,a4