LEZIONE 2

**RICORDA BENE**

* Differenza tra programma e processo:
* Un programma è scritto in maniera statica su ad esempio un hard disk.
* Un processo è un programma attivo.

Strutture dei sistemi di calcolo unità a nastri magnetici

Stampante

Controller per unità a nastri magnetici

Controller per stampante

Controller per dischi

CPU

Controller di memoria

Memoria RAM

* BUS DI STISTEMA: è un canale di comunicazione molto importante perché tutte le componenti non parlano in modo diretto tra loro, ma possono comunicare solo se migrano i dati verso questo canale (tipo linea telefonica).
* Il controller è un’interfaccia, questo sta a significare che i dispositivi possono comunicare solo mediante tale interfaccia. La necessità di questo controller risiede nel fatto che i dispositivi hanno div erse velocità di trattamento dei dati, di conseguenza per comunicare hanno bisogno di un’interfaccia che serve a compensare queste differenze di velocità, ma anche di compensare quelle che sono le regole di trattamento dei dati da parte di questi dispositivi, dato che ognuno li tratta in maniera differente, di conseguenza è fondamentale quando i dati transitano da e verso un dispositivo , che vengano trattati nel linguaggio più semplice che possa essere compreso dal bus in modo che possano colloquiare tra loro.

Funzionamento di un sistema di calcolo

* Il funzionamento è basato sul fatto che i dispositivi cooperino in **modo concorrente.**
* **NOTA BENE**

CONCORRENZA ≠ PARALLELISMO

* PARALLELISMO: riguarda dispositivi tra di loro indipendenti. Quindi non è necessario che un dispositivo per trattare i dati debba comprendere cosa stia facendo l’altro dispositivo.
* CONCORRENZA: i dispositivi sono immersi all’interno di un unico sistema e quindi dovranno tra di loro “gareggiare” per accaparrarsi le risorse.

Quindi concorrere vuol dire proprio trovare un insieme di formalismi per comprendere di chi è il turno per poter usare una determinata risorsa.

* Ogni controller si occupa di un particolare tipo di dispositivo.
* Ogni controller ha un buffer locale.
* **IL BUFFER LOCALE** è una memoria nella quale mettiamo momentaneamente dei dati in modo tale che poi il dispositivo prenda man mano i dati e li elabori (questo serve perché i dispositivi non elaborano tutti alla stessa velocità i dati).
* La CPU sposta i dati da/verso la memoria principale da/verso i buffer locali.

**RICORDA BENE**

**I dati per essere trattati devono passare per la memoria centrale**

* L’I/O avviene dal dispositivo del buffer locale del controller
* Quando un controller prende in carica i dati e li elabora, deve avvertire la CPU dell’inizio e della fine del processo, vengono lanciati dei segnali elettrici **(INTERRUPT)** di inizio e fine che viaggiano sul bus delle interruzioni.

Funzioni comuni dei segnali d’interruzione

* Un interrupt deve causare il trasferimento, il controllo di un’appropriata procedura che gestisce tutta l’interruzione.
* L’architettura deve essere in grado di gestire queste interruzioni e di salvare l’indirizzo dell’istruzione interrotta.

Questo significa che quando in esecuzione c’è un certo processo, se ad un certo punti si ha un’interruzione bisogna congelare l’indirizzo di dove si è arrivati con l’elaborazione, per poi poterla riprendere successivamente da quel punto al termine dell’interrupt.

* **IL SEGNALE D’ECCEZIONE**(TRAP) l’interruzione ed è causato da un programma in esecuzione, che può richiedere l’intervento del sistema per la gestione di questa interruzione.
* Un moderno SO è guidato dalle interruzioni **(INTERRUPT DRIVER)**. Significa che l’elaborazione avviene tramite questi segnali d’interruzione, quindi interrompendosi in continuazione si stabilisce qual è la procedura che deve andare in esecuzione o deve terminare.

Gestione dell’interruzione

* Il SO deve memorizzare l’indirizzo di ritorno nella pila (stack) di sistema. Significa che se c’è un’interruzione, quindi se si interrompe un processo che sta elaborando i dati, bisogna ricordarsi da dove riprendere quando l’interruzione sarà completata.
* Il SO deve determinare che tipo d’interruzione si è verificato:
* **POLLING**
* **VECTORED INTERRUPT SYSTEM**

Possiamo usare diversi metodi per far comunicare tra loro i dispositivi. Quello più semplice è il polling.

**POLLING:** è un’interruzione ciclica; cioè ciclicamente vengono interrogati i dispositivi (uno alla volta)

per chiedere a essi se deve effettuare una proiezione (trasferimento) I/O di dati.

* **Vantaggi**: è un metodo molto semplice perché bisogna solo creare una circolarità e interrogare singolarmente i dispositivi e fare ciò in maniera indefinita.
* **Svantaggi**: qualora ci siano pochi scambi d’informazioni, bisogna consumare tanta energia per poter fare tale ciclo. Quindi è semplice ma ha un grande dispendio d’energia.

**RIOCORDA BENE (possibile domanda esame)**

E’ sempre sconveniente il polling? NO, in quanto è conveniente quando si hanno alte situazioni di traffico, di congestione (ovvero se ci sono molte domande).

**VECTORED INTERRUPT SYSTEM:** è una tecnica d’elaborazione in cui il dispositivo d’interruzione indirizza il processore alla routine di servizio d’interruzione appropriata.

* Devono essere predisposti segmenti di codice che trattano le singole interruzioni che possono essere diversificate. Quindi il SO fa da mediazione tra CPU e le varie componenti e decide quale criterio adottare per interrogare le periferiche.

Diagramma temporale delle interruzioni per un singolo processo che emette dati

interruzione richiesta dal processo

CPU processo utente in

Esecuzione

Sta elaborando

Gestione delle interruzioni qui è basso quindi no

di I/O

dispositivo trasferimento

I/O dati inattivo

Richiesta di trasferimento richiesta di trasferimento

I/O effettuato I/O effettuato

SPIEGAZIONE DELLE PARTI:

La prima onda (riguarda la CPU) funziona così: la CPU sta trattando un processo, quando l’onda è alta il processo utente è in esecuzione. Quindi l’onda ci fa vedere quando un processo è attivo o inattivo.

Quella del dispositivo I/O quando la linea è alta il trasferimento dei dati è inattivo, viceversa quando è bassa.

Le linee verticali simboleggiano le interruzioni ossia la segnalazione che sta per avvenire un evento di scambio di dati.

COME LEGGERE IL GRAFICO Il processo parte, chiede un’interruzione (richiesta I/O), ma questa non è immediata (pezzetto dopo prima linea verticale) questo perché dopo la richiesta, il dispositivo si dispone per il trasferimento dei dati. Quando il trasferimento è completato, stesso il dispositivo lancia il segnale di fine (seconda linea verticale); e cosi continua in quanto magari dopo questo c’è un altro processo che ha qualcosa da fare.

Struttura di I/O

* Una volta iniziata l’operazione di I/O, si restituisce il controllo al processo utente solo dopo il completamento dell’operazione I/O.
* L’interruzione **WAIT** sospende la CPU fino al successivo segnale d’interruzione.

-La wait è una **system call**.

-Le **system call** sono delle vere e proprie chiamate, comandi che agiscono sulla gestione di queste interruzioni per lo scambio d’informazioni.

Due metodi di I/O

SINCRONO ASINCRONO

PROCESSO RICHIEDENTE …….. ATTESA………..

DRIVER DEL DISPOSITIVO

GESTIONE DELLE INTERRUZIONI

DISPOSITIVO TRASFERIMENTO DATI

PROCESSO RICHIEDENTE

DRIVER DEL DISPOSITIVO

GESTIONE DELLE INTERRUZIONI

DISPOSITIVO TRASFERIMENTO DATI

UTENTE UTENTE

NUCLEO NUCLEO

TEMPO TEMPO

Queste sue immagini denotano due criteri di scambio delle informazioni

-**DRIVER: software che gestisce il dispositivo per la predisposizione I/O.**

1. **IL CRITERIO SINCRONO:** i dispositivi che intervengono devono essere entrambi disponibili in quanto devono gestire il trasferimento dei dati.

**SPIEGAZIONE IMMAGINE** Quando il processo richiedente deve fare un trasferimento di dati, il kernel chiama il dispositivo mediante il **driver;** dopo di che il gestore delle interruzioni deve rendere disponibile il dispositivo. Appena questo è disponibile effettua il trasferimento dei dati.

1. **IL CRITYERIO ASINCRONO:** lo scambio dei dati non necessariamente richiede la compartecipazione dei due dispositivi.

**SPIEGAZIONE IMMAGINE** Molto simileper certi aspetti al sincrono, tranne per il fatto che in un processo asincrono l’interruzione è chiesta all’inizio, dopo di che il processo nell’attesa dei dati che sono elaborati dalla parte kernel (quarto quadrato) può fare anche altre cose, non necessariamente deve stare fermo in attesa dei dati trattati dal dispositivo e resi disponibili solo alla fine (come nei sincroni).

Tabella di stato dei dispositivi

Il SO deve conoscere lo stato dei dispositivi, ovvero deve sapere se un certo dispositivo è impegnato o meno. Questa informazione viene sfruttata per sapere se un dispositivo può essere interrogato di nuovo. La tabella di stato serve proprio a sapere se è possibile effettuare un’operazione.

ESEMPIO TABELLA

**Richiesta per la stampante**

**Indirizzo:38546 lunghezza:1372**

Dispositivo: Tastiera Stato: inattivo

Dispositivo: Stampante Stato: attivo

Dispositivo: mouse Stato: inattivo

Dispositivo: Unità a dischi 1 Stato: inattivo

Dispositivo: Unità a dischi 2 Stato: inattivo

………. ……….. ……….

**Richiesta unità a dischi 2 File: yyy operazione:**  write **indirizzo:03456 lunghezza:500**

**Richiesta unità a dischi 2 File:xxx operazione:**  read **indirizzo: 43046 lunghezza:20000**

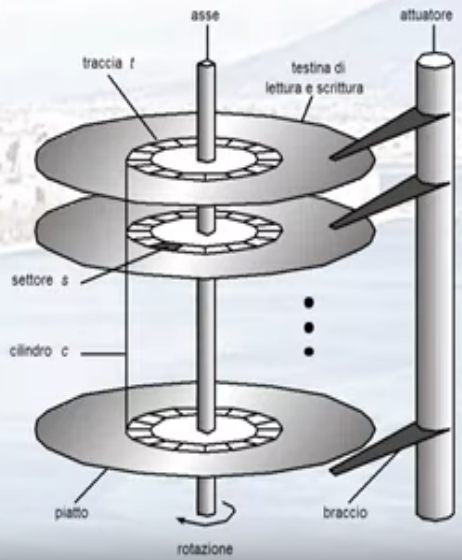
Accesso diretto alla memoria (DMA)

* Tecnica usata con dispositivi I/O particolarmente veloci.
* **Il controller trasferisce un intero blocco di dati dalla propria memoria di transito direttamente nella memoria centrale, o viceversa, senza alcun intervento della CPU.**
* Il trasferimento richiede una sola interruzione per ogni blocco di dati trasferito, piuttosto che per ogni byte (o parola).

Struttura della memoria

* **MEMORIA CENTRALE (RAM):** dispositivo di memoria direttamente accessibile dalla CPU.
* **MEMORIA SECONDARIA (HARD DISK)**: estensione della RAM capace di conservare grandi quantità d’informazioni in modo permanente (sono capaci di memorizzare informazioni anche quando la corrente è spenta).
* **DISCO MAGNETICO:** è l’unità secondaria d’informazioni più importante in quanto permette la memorizzazione perenne dei dati. In effetti l’hard disk conserva le informazioni su delle tracce, e i dati sono scritti su dei cerchi concentrici.

Schema funzionale di un disco



Gerarchia della memoria (studia bene….possibile domanda esame)

* I componenti di memoria di un sistema di calcolo possono essere organizzati in una struttura gerarchi in base a:
* Velocità
* Costo
* Volatilità
* **CACHE:** copia temporanea di informazioni in un’unità più veloce; la memoria centrale si può considerare una cache per la memoria secondaria. Piramide delle memorie in base alla velocità

**Immagine che contiene screenshot

Descrizione generata automaticamente-REGISTRI:** celle di memoria che si trovano all’interno della CPU ed è estremamente veloce. In effetti proprio perché si trova nella CPU è l’oggeto in grado di trattare i dati alla più alta velocità possibile.

**-CACHE DEL PROCESSORE:** mdemoria molto piccola che si trova all’interno del processore che permette di trattare i dati ad altissima velocità.

**-RAM:** memoria centrale anche essa molto veloce.

**-DISCO RAM:** si può configurare all’interno della RAM. Spesso si ha a disposizione più RAM del necessario, quindi se ne può approfittare per configurare una parte di questa memoria come disco e far finta di vederla in maniera virtuale. In effetti il SO assegna a un pezzo della RAM una etichetta, per esempio k, e assegna a k una logica di gestione, dal punto di vista dell’utente, simile a quella di un hard disk virtuale e i dati sono cosi trattati più velocemente.

**-DISCHI OTTICI:** leggermente più lenti dei dischi magnetici. Un esempio di disco ottico è il CDrom.

**-NASTRI MAGNETICI:** ancora più lenti dei dischi ottici anche se capaci di memorizzare una quantità di dati molto elevata.

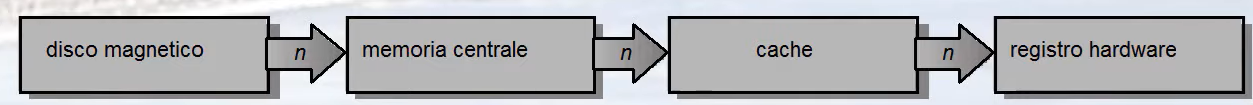
Cache

* Utilizzo di una memoria ad alta velocità per registrare dati ai quali si è avuto accesso recentemente

(e che si prevede serviranno ancora).

* Richiede una politica di gestione della cache.
* La cache introduce un altro livello nella gerarchia della memoria. Ciò richiede che i dati il quale sono memorizzati contemporaneamente su più livelli siano coerenti. **SPIEGAZIONE**

Uno stesso dato può essere memorizzato in più punti.



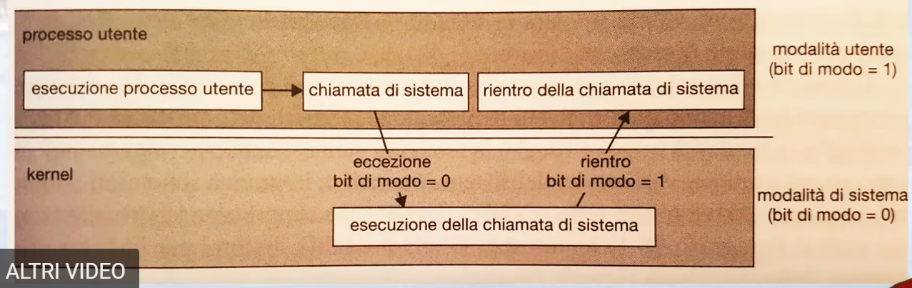
Se si ha una variabile x=4 nel disco magnetico e la si vuole modificare, succede che il programma scritto nel disco magnetico viene copiato nella RAM diventando così un vero e proprio processo. Questo per poter essere eseguito deve essere copiato nella cache del processore, poi per essere eseguiti i dati, questi volta per volta vengono copiato nel registro hardware. Il fatto che la stessa variabile la si può trovare in più punti comporta che c’è un problema di disallineamento il quale può condurre in grossa difficoltà. Se per esempio il dato x=4 che sta nel disco magnetico è stato aggiornato a all’interno del registro come x=5 e ad un certo punto manca la corrente prima che si possa aggiornare il disco magnetico, si hanno più dati che rappresentano lo stesso oggetto (x=4, x=5) il che è un problema. Bisogna perciò che ci sia una sorta di sincronismo tra questi dati nel caso in cui rappresentino lo stesso oggetto. Questa cosa viene chiamata PROBLEMA DI COERENZA DELLA CACHE.

Architetture di protezione

Servono per:

* Duplice modo di funzionamento (DUAL-MODE): la CPU tramite il SO può lavorare in duplice modo
* Protezione dell’I/O
* Protezione della memoria
* Protezione della CPU: esempio proteggerla da un uso sconsiderato di un processo

DUAL-MODE All’interno della macchina ci sono sia programmi del SO, sia programmi degli utenti. A differenza del programma utente che si può anche interrompere e avere un malfunzionamento, quello del SO deve funzionare ed essere efficiente, quindi in qualche modo bisogna conoscere attraverso una variabile se in un certo istante all’interno della CPU si sta lavorando a delle istruzioni che appartengono a un utente oppure se tali istruzioni appartengono al SO. Quindi esiste questa duplice modalità di funzionamento che impiega il **BIT DI MODO** che vale 0 se si stanno eseguendo funzioni di sistema, oppure 1 se si stanno trattando istruzioni dell’utente.

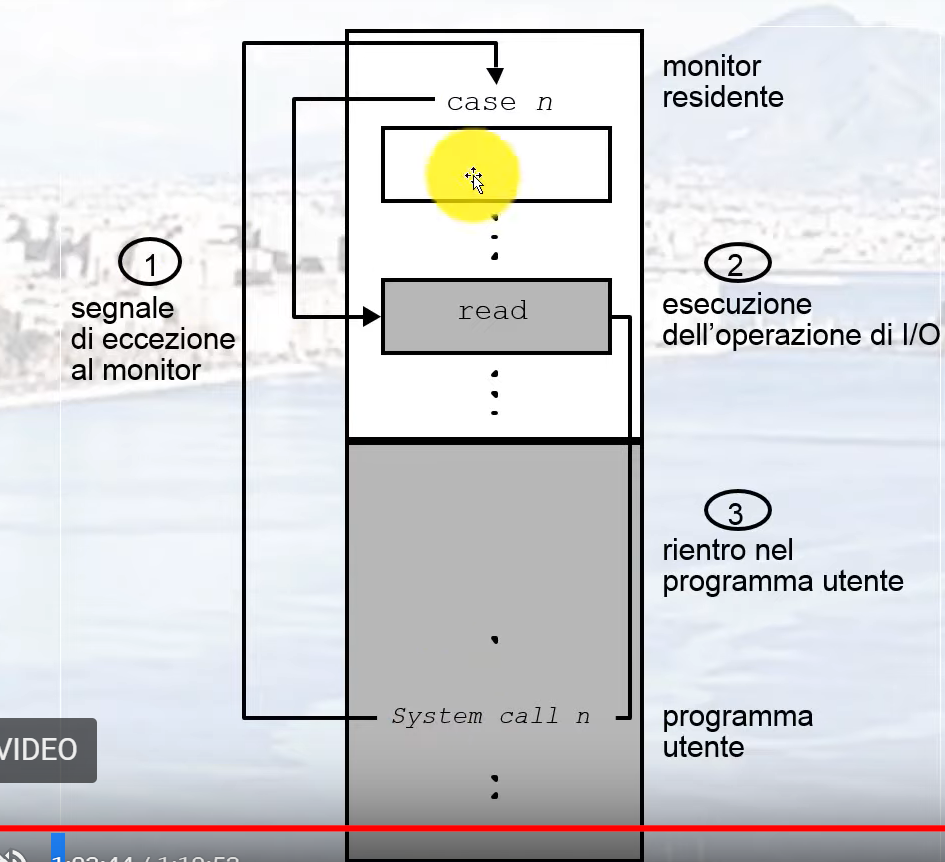


* **BIT DI MODO:** indica quale modo è attivo: di sistema (0) o d’utente (1).
* Ogni volta che si verifica un’interruzione o un’eccezione si passa dal modo d’utente al modo di sistema, cioè si pone a 0 il bit di modo.****

La CPU consente l’esecuzione di **istruzioni privilegiate** soltanto nel modo di sistema.

Protezione dell’I/O

* Tutte le istruzioni di I/O sono istruzioni privilegiate
* Affinchè la protezione dell’I/O sia totale, è necessario evitare che l’utente possa in qualsiasi modo ottenere il controllo del calcolatore quando questo si trova nel modo di sistema.

**USO DI UNA CHIAMTA DI I/O**

**SPIEGAZIONE** Si è nel programma utente; questo ad un certo punto fa una system call e questa automaticamente effettua una chiamata con una qualche opportuna procedura mediante un segnale di eccezione al monitor (1). A questo punto il controllo non è più dell’utente ma del monitor residente il quale controlla chi ha causato questa eccezione (case n) per poter poi decidere la procedura con la quale trattare l’eccezione. Dopo che questa viene eseguita viene fatta un’operazione d’I/O (2) e il controllo ritorna di nuovo al programma utente il quale proseguirà con la sua esecuzione (3).

Protezione della memoria

* E’ necessario fornire protezione della memoria almeno per il vettore delle interruzioni e le relative procedure di servizio contenute nel codice del SO.
* Questo tipo di protezione si realizza impiegando due registri che contengono l’intervallo degli indirizzi validi a cui un programma può accedere:

-**REGISTRO DI BASE:** contiene il più basso indirizzo della memoria al quale il programma dovrebbe accedere (dove inserire il job.

-**REGISTRO DI LIMITE**: contiene la dimensione dell’intervallo (quanto è lungo il job).

**USO DI UN REGISTRO DI BASE E UN REGISTRO DI LIMITE**

**Immagine che contiene screenshot

Descrizione generata automaticamente**

Architettura di protezione degli indirizzi

Immagine che contiene testo, segnale

Descrizione generata automaticamenteTra la CPU e la memoria RAM ci sono una serie di meccanismi che permettono la traduzione di un indirizzo e che inoltre serve anche a proteggere. In realtà la CPU lavora con degli indirizzi che sono virtuali, cioè se si scrive un programma e questo ha una locazione di memoria d’inizio =123000, tale indirizzo è virtuale, ovvero nella RAM l’indirizzo non sarà quello perché solo un valore virtuale che fa parte solo del vettore/array/matrice usate nel programma. Mentre nella RAM gli indirizzi sono fisici

**FUNZIONAMENTO**

L’indirizzo della CPU, per sapere se è legittimo, viene confrontato per protezione con la base; se risulta maggiore va bene; ma ciò non basta perché deve in realtà essere compreso tra la base e la base+limite.

Protezione hardware

Avviene grazie ad un meccanismo che permette solo al SO di accedere direttamente al dispositivo, a differenza dell’utente che non può farlo.

Protezione CPU

* **TEMPORIZZATORE (TIMER):** invia un segnale d’interruzione alla CPU a intervalli di tempo specificati per assicurare che il SO mantenga il controllo dell’elaborazione.
* Il timer si decrementa ad ogni impulso.
* Quando raggiunge il valore 0, si genera un segnale d’interruzione.
* I temporizzatori si usano comunemente per realizzare la partizione del tempo d’elaborazione (time sharing).
* Un altro dei temporizzatori è il calcolo dell’ora corrente.
* **Load-timer è un’istruzione privilegiata**