

Sistemi operativi anno 2021-2022 registrazioni: 2° semestre

2022-02-05

slide 00-intro.pdf

Terzi abbiamo visto queste delle cose abbiamo visto cosa abbiamo fatto. Il Sistema Operativo è un livello di astrazione e d'importante è che fornire un API in maniera tale che il processo intero venga confinato controller in maniera tale che il processo corrente possa, anche qui c'è la metafora con il mondo reale, convivere civilmente con processi che condividono l'ambiente di lavoro. Il Sistema Operativo è un gestore di risorse, rende tutto più facile. Macchina Estesa, dipende dall'hardware, rende i programmi nostabili. Alliamo lavorato nello storia, voglio sommare qui (allo storia) un ultimo perché "cioè" che avviene in processi non è detto che non sia utile anche nel futuro". Questa (00-intro.pdf pag 11 Generazione Z (1955-1965)) che sembra un'architettura antica, decisiva in realtà è usata ma non in sistemi general-purpose per esempio chi di voi ha giocato con un ordinatore da che non sa può dire che abbia un sistema operativo, ma proprio un monitor oppure chiamate browser per l'ordinatore. Cioè l'ordinatore altro non è che un microcontrollore, c'è anche l'ordinare con il nostro chip dello raspberry pie, comunque quello classico è un al-mega

di qualche genere diciamo per molti arduini è il 328 e che sarebbe un debole sistema per inserire, per caricare il programma avrebbe bisogno di un device esterno. L'arduino è di fatto un microcontrollore con in più un loader, un loader che viene precaricato, che viene mantenuto in uno eeprom (più che in una eeprom in una non volatile ... una eeprom). Arduino al mega non è una macchina di von Neumann è una macchina di Harvard, quindi ha due memorie separate, una per i programmi e una per i dati. Nella memoria per i programmi che è

non volatile, per essere un arduino viene precaricato un piccolo programma, che può essere simile a questo monitor, che all'accensione o al reset per pochi secondi si ferma e guarda se sta arrivando dati dalla seriale emulata su USB e se in questi pochi secondi arrivano dati li interpreta come un programma da caricare sempre nella pronta memoria. Oltre il loader. Quindi associglia molto a questo (ges. 2) vedete il compilatore Esercizi se voi avete l'arduino e il primo programma tipico dell'arduino blink, viene caricato blink e fa quello che deve fare poi resetta / riaccende la / carica, un altro programma e fate altre cose. Quindi come vedere queste architetture ritengono in altri ambienti, non si può dire che sia un sistema operativo

è molto emozionale come sistema.

(va a pag 37) dicendo che abbiamo visto le cose in maniera...)

Ecco allora visto tante funzionalità che possono avere i sistemi operativi. I sistemi operativi si riconoscono dall'interfaccia delle system call dalla lunga API. E allora visto che ci sono delle funzionalità che ci possono essere o non essere e che se ci sono o non ci sono si evince dal fatto se l'interfaccia delle sistemiche di questi sistemi prevede o meno funzionalità chiamate specifiche per queste funzioni. Se manca tutta la classe di sistematiche per definire chi è l'utente del processo corrente e cambiare l'utente del processo corrente il sistema non sarà multiuser, ma sarà pensato per fare programmi senza protezione tra utenti e così via. Oltre alle cose classiche ci sono altre funzionalità del sistema operativo che possono esserci o non esserci e queste caratterizzano sistemi operativi per usi specifici per esempio i sistemi operativi personali soprattutto o no il PARALLELSIMO (pag 37). Cosa è il parallelismo? Lui dice la parola, la capacità di fare più operazioni contemporaneamente. Un processore standard normale, un singolo CORE, fa una cosa sola in un istante di tempo. Quindi abbiamo visto che i sistemi operativi capaci di funzionare su singolo processore o eseguire codice del KERNEL o eseguire codice applicativo e possono darli uno all'altro o da kernel a user.

caricando uno stato o da user a kernel perché avvienne una richiesta del processore a un altro evento avvenuto. Come possono essere i sistemi paralleli? I sistemi paralleli possono essere Single Instruction Multiple Data (SIMD) detti anche sistemi vettoriali. Quindi questi sistemi fanno più cose contemporaneamente, ma in realtà eseguono la stessa istruzione su dati diversi, su rettangoli su sequenze di dati in modo che l'esecuzione venga fatta nello stesso momento su tanti dati. E se no ci sono le Multiple Instruction Multiple Data (MIMD) che sono quelle che eseguono programmi differenti su dati differenti. Quindi per esempio se avete un sistema multicomputer, un sistema multicomputer è una macchina parallela di tipo MIMD. E voi conoscete dei sistemi SIMD di avere mai visto? Pensate che sono dei processori che non come sono fatti velocizzano le operazioni che devono essere fatte contemporaneamente su grandi quantità di dati. Non vi rilievo finemente un esempio? Bravi!
Studente: Forse nella lavorazione delle immagini sui Tanti pixel

Sì, è chi fa elaborazione delle immagini gestendo Tanti pixel? Le SCHEDE VIDEO, e infatti queste schede video noi solitamente venivano chiamandole con il nome di GPU (Graphics Processing Unit) per fare elaborazioni dove lo stesso operazione deve essere fatto su grandi quantità di dati. Per esempio l'implementazione

della evoluzione di una rete neurale. Per cui abbiamo un settore che ci dà tutti gli dati di eccitazione dei vari neuroni, abbiamo la matrice con i pesi che ci indica quanto lo stato di un neurone debba proporsi allo stato di un altro e per ovviare la rete bisogna che questi calcoli vengano svolti su tutti i neuroni che compongono la rete neurale. E quindi una macchina SIMD è molto comoda per questo. Un altro modo di classificare machine parallele è quello di guardare quanti CORE / processori ci sono. Ci possono essere sistemi a base parallela, pochi processori in genere molto potenti, o sistemi massicciamente paralleli, un gran numero di processori che possono anche avere avere potenza non elevata

(pag 38)

ma altri tipo di classificazione dei sistemi paralleli sono sistemi TIG-I TTL Y COUPLED, quindi sono sistemi dove in core collaborano all'interno di un architettura che prende il loro di comunicazione a memoria comunitaria. È questo il modello dei sistemi Personal Computer Multicore, dove avete ... che ne so ... 2, 4, 8 core però questi CORE condividono il bus, la memoria. Ok? È invece sistemi LOOSELY COUPLED dove sistemi dove ogni unità di elaborazione è un processore con la sua memoria (memoria privata), e i suoi canali di comunicazione ed è collegato con altri processori secondo uno determinista TOPOLOGIA. ok? Normalmente si ottiene: Highly

coupled usano prochi processori, si sistemi loose (y coupled usano lontani processori. Perché secondo voi non ci fanno sistemi tightly coupled con 256 processori / core ? Bregoli studente: Perché più aumentano i processori più sarà il numero di processori che nello stesso momento chiederanno l'accesso alla memoria e quindi

Esattamente ! Perché essendoci un bus a memoria comune diventa collo di bottiglia. Quindi: non ci ottengono prestazioni perché diventa elemento critico il bus perché è vero che ci può rendere di più e tentare di ottenerne dei bassi più performanti. Più veloci però c'è un limite fisico alla cosa. Mentre invece ci sono state sperimentazioni per esempio la CONNECTION MACHINE era una macchina che aveva 65.536 processori collegati fra loro con una topologia ad PERCUBO. Che cos'è un CUBO ? "Non ci capisce un culo" :D no... (Carriaggio) ...

dimensioni	1	2	3	...	N
cubo	• - •	! - !	! - !	! - !	• - •

e noi potete continuare !

Questo è un cubo. Ma questo è un particolare cubo.
E il cubo di dimensione 3. Esistono cubi di TUTTE le dimensioni. Il cubo di dimensione uno. • Cubo di dimensione 2 □. Cubo di dimensione 3 ☐. E poi notate continuare. Come potete creare il cubo di dimensione $N+1$? Per fare il cubo di dimensione $N+1$ prendete 2 cubi di dimensione N e collegate tra loro i modi corrispondenti. Quindi qui abbiamo un cubo di dimensione 1 •, qui me ho presi 2 : i: e li ho collegati assieme !!. Qua ho preso 2 cubi di dimensione 2 !: i: e li ho collegati assieme !!. Se volerai passare alla dimensione 4 già ancora di più vedere nello spazio fatta due cubi concentrici uno dentro l'altro e collegate i nodi. Ok, Questo è correlato ai numeri binari. Quis. — avete un bit o 0 o 1 Qui !: avete fatto due copie e quindi il primo bit vi rappresenta a quale copia volete accedere e il secondo bit, il bit meno significativo è il bit all'interno del cubo precedente. Per fare un cubo in più il primo bit vi indica a quale cubo del livello precedente scegliete e noi avrete l'indirizzo nel cubo più basso. Quindi collegando i processori secondi un PERCUBO con 65·536 punti (infatti dire che è un cubo di dimensioni 1 6) la cosa della è che ogni processo

$$2^{16} = 65\,536$$

può parlare con ogni altro processore facendo non più di 16 passi, nonostante ci siano 65.536 processori. La connection machine è fallita, perché secondo voi? L'idea della connection machine era quella di prendere dei processori a bassa potenza e ne mettene insieme dei numeri massimi. (un esempio di loosely é la connection machine). Anche se un processore non é una valvola, anche i processori hanno la loro possibilità di guastarsi, e mettendone insieme 65.536 uno ha la probabilità di guadare un processore 65.536 volte quella di un processore singolo. Questo é uno dei fattori e l'altro é mentre con un sistema / i sistemi che abbiamo i nostri portatili che hanno N core questi sistemi vengono utilizzati spesso anche con programmi che non sono adatti all' elaborazione parallela, ma perché avere bioghi di fare più cose quindi mettere un processore che non è parallelo che funziona su un processore, mentre un altro funziona in quell' architettura per utilizzare efficacemente delle architetture di questo genere tutti gli algoritmi che andate a scrivere dovranno essere pensati per funzionare là sopra. Perché non ha senso mandare li sopra l'esecuzione di un processo sequenziale inferocchio standard.

Perché userà un processore e me anche e andrai piano come un singolo processore. Ok? Quindi comunque esistono sistemi Crossley coupled

anche modifichi con meno processori e ... o forse oppi rimasconi ma rimasconi nella forma di SISTEMI DISTRIBUITI (cap 40 noi toriamo a 39).

Cioé invece di pensare a una macchina singola con tanti processori si dice "prendiamo tanti sistemi come sistemi singoli: (con processore, memoria ... come tanti personal computer) e poi facciamo in modo di considerare tutte queste insieme di machine convenzionali: come un sistema unico". ok? Ed è l'architettura che stanno usando per fare i nuovi supercomputer. avete sentito Leonardo? Se poi andate anche molto più vicino all'INFN in area Moro Ronti (ci si arriverà a piedi) avete in mente dove è il dipartimento di fisico? Se andate a vedere i siti online ci sono sistemi di elaborazione che si fissaano tantissimi per fare elaborazioni delle tracce degli acceleratori, elaborano dati che vengono dal CERN. Vedete che lì avete distese e distese di "rack" con nei "rack" dei server, ma quei server non sono dei server singoli: quei server vengono usati come un unico sistema di elaborazione che coinvolge tutte queste macchine. E quindi se volete far differenza tra sistema massicamente parallelo (totally coupled) e sistema distribuito è una differenza facile, il concetto è simile, solo che mentre

si pensava di fare una topologia, una struttura, una machine specifica nei sistemi distribuiti vengono viste tante machine convenzionali come un sistema unico. Qual'è il problema? qui (connection machine) veniva studiata una rete specifica per l'elaborazione, in questo caso tutta la topologia di collegamento, la parte di connessione tra processori viene fatta con sistemi specifici di rete. Qual'è il problema nei sistemi distribuiti? Siccome vengono sistemati di rete e non delle architetture di bus interne come quelle loosely coupled, hanno maggiore latenza e questo si ripercuote sugli algoritmi e sullo studio di come sfruttarli appieno. Questo è più o meno una descrizione hardware di come sono fatte le machine. Come ammatta tutto questo mio sistema operativo?

(nag 39) Parliamo di sistemi paralleli, sistemi multicore, sistemi tightly coupled. I sistemi operativi più diffusi vengono MULTi PROCESSING SIMD ET TRICO (SMP). Cosa significa che tutti i processori vengono gestiti nello stesso modo. "Non c'è un processore più grande degli altri" citando Truxell. Tutti i processori eseguono una copia identica del sistema operativo e eseguono processi. Quindi qual'è il problema? Se abbiamo già in parte risorse, che questi vari processori vivono come il processore singolo quindi eseguono processi, succede una cosa, viene chiesta una system call, arriva un interrupt, vuole eseguire il kernel, esegue il kernel e fa Dunque il problema è che se ci sono più processori

che vogliono eseguire il Kernel, questi processori hanno strutture dati in comune. E quindi occorre fare in modo che in coordinamento nell' accedere alle strutture dati. E', ripassando dal primo semestre, siccome i vari processori hanno disponibilità di: interrupt indipendenti occorre utilizzare altre davoerie, altre soluzioni per creare sistemi di mutua esclusione, ed è per questo che sono nati gli spin-lock. Quindi per fare dei sistemi operativi SMP occorre che i processori forniscano delle istruzioni per consentire spin-lock, quali test & set, atomicsw op., e la copia di istruzioni lck-load store-conditional che è la copia di istruzioni che consente di fare test & set su processori RISC. I processori RISC non possono fare due cose in un colpo solo quindi sono due istruzioni, ma sono fatte in maniera che la seconda fallisce se è successo qualcosa tra la prima e la seconda. L'altra alternativa è MULTIPROCESSI, NO ASIMMETRICO quindi si può pensare di avere un processore master che gestisce il Sistema Operativo, gestisce l'assegnamento dei processi ai vari processori, e gli altri processori sono solo slave che eseguono i singoli processi come viene indicato dal processore master. (nog 4) SYSTEMI DISTRIBUTI invece si comportano in modo diverso, appaiono come una unica macchina. Un esempio di Sistema Operativo Distribuito lo potete pensare come quelli del nostro laboratorio. Se accedete al laboratorio esistono tante macchine, le riconosce la rete di fianco al login di ogni macchina

c'è scritto il nome alla quale vi colleghete. Sono nomi di personaggi d'opera... Però quando accedete (anche remotamente via SSH) alla fine non vi interessa niente a quale macchina voi avete fatto accesso. Il sistema è tutto allineato. Avete la nostra home directory doveunque voi state (non come i sistemi delle aule, in cui abbiamo una home directory staccata in ogni aula quindi le nostre preferenze...). E ciò che non vedete è che in zone dei sistemi di istallazione, aggiornamento di tutto il sistema distribuito che sono condivisi. Quindi se c'è da aggiornare il sistema non è che in tecnici del nostro dipartimento vadano di macchina in macchina ad aggiornare il sistema, ci sono dei meccanismi per tenere tutto il sistema distribuito allineato con quello che serve. Quindi in questo caso quelli che ci servono dal punto di vista dei sistemi Operativi per dare questa situazione è ovviamente condivisione efficace del file system, poi occorre datalabare condivisi per quello che sono elementi, password, gruppi e così via, possono esercizi meccanismi per distribuire il carico fra computer se volete utilizzarli per fare elaborazione massiva/massiccia. (pag 42) Andiamo in un'altra classe di sistemi in realtà poco conosciuti, i sistemi REAL TIME, penso che abbiate sentito questo termine solo che di solito non vengono definiti correttamente. REAL TIME non significa veloce, può essere tranquillamente lento, dipende dai problemi che dovete risolvere. La definizione esatta di sistemi real-time ha trovato qui (pag 42):

o Dedicati senza disponibilità sulla pagina del lessico.

Definizione: Sistemi real-time

Sono i sistemi per i quali la correttezza del risultato non dipende solamente dal suo valore ma anche dall'istante nel quale il risultato viene prodotto.

Vi dicono in sistemi real-time sono i sistemi per i quali il risultato è corretto non solo se è corretto nel proprio valore ma se è corretto nel tempo in cui viene prodotto. Ok? Quindi tecnicamente "se is project un sistema per dire... deve trovare la risposta alla vita, l'universo in ogni cosa ma non ci deve mettere meno di 200 anni" è real time. Perché se quello mi risponde a 199 anni la risposta è sbagliata perché gli avrò creato un vincolo temporale, sia chiaramente questo è uno scherzo con citazione di Douglass Adams, ma ovviamente quello che interessa è avere un risultato entro un determinato tempo. Però non è detto che sia... l'importante è che sia deterministica, non ci interessa che sia veloce, che sia abbastanza per il problema che andiamo a risolvere. (frag 43) Infatti abbiamo due tipi di sistemi real-time: HARD real-time e SOFT real-time. Quelli hard real-time sono quelli con effetti catastrofici: se il vincolo temporale non viene rispettato. Ruritopno in questi giorni è tornata per vicinanza geografica alla mente il problema della centrale di Chernobyl. Una centrale nucleare ha un sistema di controllo, un sistema di controllo che come pensate è un sistema realizzato

con dei sistemi di elaborazione quindi materia nostra.

Questo sistema non è noi nelle centrali a fissione... non è poi un sistema tanto elaborato perché ci sono dei meccanismi di rallentamento della reazione nucleare e quindi se la temperatura diventa troppo elevata occorre limitare, occorre ridurre la velocità della reazione. Nelle vecchie centrali avevano delle barre di grafite che venivano messe, inserite o estratte, dal nucleo di uranio. Ora però queste operazioni fatte nell'arco di dieci minuti, anche... non so esattamente in tempi massimi di millisecondi, microsecondi. Quindi anche un drolino potrebbe farcela per sole fare il calcolo necessario però deve essere garantito, reached, se o non avviene in tempo perché si blocca il sistema oppure non hanno previsto che poteva arrivare l'acqua del maremoto come Fukushima e gli effetti sono catastrofici ma non sono le centrali nucleari gli unici ambienti. Se avete visto, ormai da lontano, gli apparati della terapia intensiva, sono appositi con a bordo degli elaboratori. Non è il caso che questi elaborano le tempestiche delle loro attività, che gli infusori diano troppo o troppo poco farmaco o ai blocchi tutto quello che qualcuno dia control + Alt + Delete, perché alla macchina si può fare mai altro persona collegati NOI. Non solo non se pensate agli dei dei, ai grandi, serii di linee, spero che non

menziate che la cloche del pilota sia collegata con dei carri agli alletttoni. La cloche non è altro che un sofisticato joystick attaccato a dei sensori che danno input a un sistema di elaborazione, ed è il sistema di elaborazione che poi pilota tutte le superfici di volo. Tutti questi sono esempi di sistemi hard real-time.

I Sistemi Operativi per i sistemi hard real-time sono completamente diversi da quelli visti fino ora.

Anche i processori dei sistemi hard real-time non sono neanche scritti in C, sono scritti in linguaggi fatti apposta dove per esempio ogni loop ha un numero massimo di iterazioni consentite se non si chiama la routine di emergenza, perché non ci si può schermare e tra l'altro non sono sistemi operativi in cui dici adesso lanci un processo, d'insieme dei processori che quel sistema deve elaborare viene stabilito a priori, viene calcolato lo Scheduler statico in modo che siamo sicuri che tutti facciano quello che devono fare, e lo si fa in condizioni non critiche, quando lo centrale nucleare è ferma, quando il sistema di controllo della terapia intensiva non ha nessun malato collegato, quando l'aereo è a terra. A quel punto si carica tutto, si mette in funzione, magari si mette in funzione anche con sistemi replicati per fail tolerance e si lancia tutto il sistema così com'è: ok?

Quelli che in realtà sono molto più diffusi sono i sistemi soft real-time. I Sistemi Operativi per Personal Computer

sono tipicamente Soft real-time. Quindi se avete un processo che sta facendo rendering di un frame video, o ancora peggio di frame audio, che sono più fastidiosi gli audio del video, quelli con un sistema di priorità si fa in modo che barattamente possono sbagliare la loro tempestività. Però capite bene che è una gamma di problemi completamente diversi. Tipicamente i sistemi soft real time sono best-effort cioè se voi prendete un sistema fatto benissimo, studiato con cura e lo sverraccicate state tranquilli che il vostro frame salta un po' e l'audio è singhiorzante. Comunque mi preme dirvi che real-time non è l'esecuzione veloce.

Così abbiamo finito i lucidi che abbiamo iniziato ieri (OO-intro.pdf (registrazione: 2022-02-25 pt 1 durata 45:40)) e cominciamo a ripassare l'architettura (01-arch-hw.pdf) in particolare guardiamo per la nostra da architettura che ci interessano per lo sviluppo dei sistemi operativi. Il segnale https://randomgame.com questo è uno gioco a livello: che cosa è l'architettura degli elaboratori, come dicono i miei: Figli...? "To ho incorniciato il gioco e sotto stava sopra tutto il processore perché non riuscivo più a sbucarmi." Quel-e ha messo giusto? "Invece? Sei arrivato a costruire completamente quello gioco logica a un elaboratore capace di

eseguire programmi. Dikensi: Se si dice *processando*
lo sarebbe *processato* (non fatto adesso se non non
stesse attende alla loro esecuzione).

bot: hard & Tetrice

prof: non ha capito la lotta tra

Questa (pag 2) è la nostra *benemarota* macchina di Von Neumann.
Forse si può dire "nirvana." delle gioco. (registrazione 48:20).

C'è la CPU che ha all'interno le unità di controllo e
l'unità aritmetico logica. C'è un bus attaccato al bus ci
sono non i dispositivi ma i controller dei dispositivi e
c'è la MMU (Memory Management Unit \Rightarrow unità di gestione
della memoria) che collega alla memoria Principale.

Sai conoscere il libraio mandato da Tetris, se volete mandare
da Tetris è molto più professionale e usa linguaggio
di descrizione di hardware. Voi avete usato sono
da Tetris al corso (di architettura)? Il gioco mi
consente di portare cose create mentre formalizzate da
quelle di mandare da Tetris anche a studenti della
scuola secondaria (\Rightarrow 1^o grado medie). Io grido dici: it's...). Ok.

(pag 3) Avete visto l'anno scorso architettura dei processori,
concetti: base relativi alla memoria e linguaggio per assembly
Avete visto un linguaggio assembly? Studente: quello di mandare da Tetris.
Concetto: comunque bene o male, come effetto collaborale,
quest'anno vedrete l'assembler di MIPS. Quando vedrete
la esecuzione il programma vedrete la transcodifica

delle istruzioni mips. Tra l'altro il prezzo che fa il reverse-assembly delle istruzioni mips è proprio un prezzo della macchina. Mamps che ho fatto con le mie manine ::D. ok.

(Cap 4) INTERRUP ! Questo è un concetto chiese fondamentale e ancora oggi vedo che spesso anche all'esame viene sbraitato.

Questa **INTERRUPT** è un meccanismo che permette l'interruzione del normale ciclo di esecuzione della CPU.

La CPU (Central Processing Unit) è quasi mai appello alle conoscenze dell'uomo scorsa, miracolando se qualcosa non è chiaro formo fermo e rivediamo tutto quello che c'è da rivedere. Le CPU vivono la loro vita caricando l'istruzione corrente, e come fanno a caricare l'istruzione corrente? Hanno un registro che si chiama Program Counter o Instruction Register, sono sinonimi. Lo truttano sul bus accendendo il bit del leggi. La memoria risponde mettendo sul bus dati l'istruzione. L'istruzione viene caricata nel registro di decodifica. A quel punto, passo successivo, il processore analizza l'istruzione caricata e (lo vedete benissimo anche su Randotronics, o con l'architettura dei libri di Tannbaum) a quell'ora che farà configurazione vari componenti per fare / eseguire l'istruzione voluta. Se pensate all'architettura di un processore, i vari bus interni al processore mi piace vedereli come i binari dei modelli ferroviari.

"Dove vanno i dati?" "Dove vanno i dati?"

dipendono da come sono stati gli scambi.

Praticamente la decodifica dell'istruzione avviene in modo che dalla stringa dell'istruzione vengano eletti gli scambi dei bus della CPU perché al passo successivo non essere eseguita l'istruzione nel caso alla fine l'istruzione fa lo store dei risultati, e così via.

Quindi come il motore a 4 tempi: aspirazione, compressione, accensione e scarico questa (CPU) fa:

carica l'istruzione, decodifica l'istruzione, esegue l'istruzione e poi torna da capo. Quando arriva in fondo e vuole tornare da capo fa un passaggio in più. Dice: "E avvenuto un INTERRUPT", o meglio l'interrupt alla fine arriverà al processore come un filo o più fili. Uno di questi bit è a 1 e sono tutti a 0? Se vede che c'è un interrupt il processore è organizzato per prendere l'indirizzo prioritario, che è stato messo apposta per questo funzionalità e metterlo nel program counter. Quindi in realtà il processore al ciclo successivo avrà fatto un salto a una indirizziata. Il programma non ha detto in quel momento di fare il salto, ma siccome è avvenuto un intervento viene cambiato il program counter, ovviamente il valore precedente del program counter lo deve salvare da parte se non riesce a tornare indietro.

Se gli INTERRUPT sono MASTERSATI questa operazione non viene fatta. E quindi anche se ci sono

questi fili accesi per dire "guarda che c'è un interrupt" il processore continua con l'istruzione successiva, e cioè.
Perché vengono introdotti gli interrupt?
L'interrupt fondamentale, l'interrupt per il quale è stato creato il concetto di interrupt è l'interrupt di Input / Output (I/O). Ricordate che gli interrupt sono quelli veri INTERRUPT HARDWARE sono comunicazioni che vengono dati controllori dei dispositivi al processore. Ok? Quindi il caso più tipico, l'alibiante visto anche con la funzione scénica, fa centralizzazione di inizio corso, l'interrupt di I/O più importante è quello che viene mandato dai controllori per dire "ho finito l'operazione". E secondo così il Sistema Operativo può attivare task operazioni di I/O nei vari controllori e poi mettersi a fare altre cose perché guarda uno dei dispositivi avrà finito l'operazione mandarò l'interrupt. Più di interessarsi del fatto che ci sono operazioni di I/O in corso perché verrà PROTATA mentre AVVIERTITO preventivamente e dimostrate, nel momento della fine delle operazioni di I/O. Ma ho domanda biondo:
"Sai se un interrupt si può fare multiprogrammazione?"
Si, ma lo farò inefficiente. Perché l'unico modo che c'è per vedere se l'unità di I/O ha finito è chiederglielo. Via l'os il processore può sempre dire "Hei! Unità di I/O hai finito? Cosa stai facendo?" Ma facendo così si dovranno utilizzare solo tecniche di POLLING.
"Hai finito?" "Hai finito?" "Hai finito?" ... In un sistema monotask, quando mette micro-sleep nello-

ordinare per fare il bit di accendi / spegni, quello prende tempo; quello sta lì a ciclare a dire "hai finito?" "Il bit finito?" ... non ha altro da fare! Ma se io usassi dei meccanismi del genere (polling) su processori da Personal Computer ovviamente perderei un sacco di prestazioni. O prendo prestazioni, chiedendo troppo process "Hai finito?", e perdono prestazioni, cioè mi accorgo dopo del tempo che l'unità di I/O ha finito alla fine perdere prestazioni lo stesso. Ecco gli INTERRUPT possono essere sia hardware che software, se volete in molti autori, anche a me piace chiamare gli interrupt hardware INTERRUPT, gli interrupt software TRAP.

Gli interrupt software è il meccanismo per l'idea di uscire lo stesso meccanismo degli interrupt per poter codificare degli eventi causati dal processo in esecuzione. Quindi occorrerà una maniera per poter fare in modo, se ho detto anche ieri, che il process si esecuzione viva in ambiente con finiti e controllati.

... far in modo che se il process in esecuzione tentava di fare una corbelleria DAVOLERIA ... d: divisione per zero, accesso in memoria dove non può uscire di istruzione illegale ... inventarsi tutto quello che può fare un process di male ... Non sopravvive tutto, ma attirasse l'attenzione del Kernel (del S.O.). Allora l'idea è stata "noi abbiamo già un meccanismo che serve per attivare l'attenzione del

Sistema Operativo quando avviene gli interrupt del controller, gli interval hardware. Utilizziamo lo stesso strumento per gestire quando succede degli errori, delle situazioni anomali causate dal processo in esecuzione. OK? Quindi: gli interval hardware sono generati dai controller e normalmente sono attività asincrone che non sono relative al processo in esecuzione in questo momento. Gli interval software invece utilizzano lo stesso meccanismo degli interrupt per errori del processo in esecuzione.

(pag. 5)

bot: quindi quando un programma interrompe la sua esecuzione per core dumped ad esempio l'esecuzione viene terminata da una trap?

Sì, se c'è segmentation fault trabatto allora...
non è di fare un del programma che prende un indirizzo 0, assegna un puntatore a 0 e tenta di stampare ciò che c'è a quel punto? Non

Tro l'altro ho notato, cosa sei lo ricordo così, no ho visto che Maps usa il valore di Null 1 a -1, secondo me quelle belle casella e lo riserviamo a 0 perché è quello standard voluto dal C. Secondo me le casella 0 la costante a ricompilare tutto sbagliabile. Funzionare tutto ugualmente, penso che succeda niente. Se cambiano la fase 1 e fare 2 così vediamo che tutto funziona. Dicono segmentation fault lo

causa il processo in esecuzione guinol "io sono il processo in esecuzione. Sono in user-mode, posso fare tutto quello che mi consente le istruzioni aritmetico-logiche e l'accesso alla mia memoria. Faccio questa istruzione in cui tento di accedere alla memoria con indirizzo sbagliato. Il ciclo di CPU carica decode esegue, l'esegui crea la trap. Al passo successivo viene generata la trap e come se fosse un interrupt si prova all'esecuzione di un indirizzo del Kernel, che quindi incomincia a... Il Kernel si risveglia praticamente, normalmente è dormiente e viene svegliato quando avviene un interrupt per una trap. Il Kernel provate, guarda quale che è successo, dice ah guarda "il processo X ha tentato di accedere..." e lo vede guardando i registri della CPU. A quel punto dice "cosa devo fare adesso?". Ed è il Kernel che mette in piedi, parla con gli UNIX, quel meccanismo di mandare al processo il SEGNALE. Quindi il Kernel dice ah, siccome io sono UNIX e c'è scritto nelle regole che per mandare "tu sei un processo for un errore di indirizzamento nella memoria devi dare segmentation fault" il Kernel chiama quella funzione per mandare un SEGNALE quella chiamata dalla Systemcall Kill di un altro processo. L'esecuzione non viene terminata dal trap, il passaggio è più lungo, il trap causa la generazione del segnale.

Non è detto che una segmentation Fault termini il processo.
Voi potete mettere nel processo un gestore di segmentazione Fault che comandi le cose e vi faccia continuare l'esecuzione.
Perché può essere causato il segmentation Fault. Il fatto che poi il segmentation Fault non venga catturato, si comporta come qualsiasi altro segnale non gestito che prende la terminazione. Prego!

Studente: Non mi è chiaro come viene generato la trap?
Perché quando... in caso di accesso in memoria la CPU esegue soltanto un'istruzione, ci vorrebbe un lato software che coprisce che quell'indirizzo è sbagliato.

Sì, sì è la MMU che comunica: È ancora uno più complesso di così, ci abbiamo un capitolo di gestione della memoria fatto apposta per questo... provate a dire...
Quando si scrive sul bus un indirizzo e alla MMU è stato indicato quale tabella delle pagine... quale risoluzione usare in questo momento. La MMU guarda in una cache, che si chiama Translation Lookaside Buffer (TLB), non se è stata visto ad architettura, c'è una piccola cache che tiene le ultime risoluzioni fatte, in realtà a livello di pagina/pagine non a livello di istruzione, comunque se lì non c'è la MMU manda una trap al processore in ogni caso. Possono darsi due casi.

Più semplicemente essere che, siccome la MMU non può tenere tutta la mappa della memoria in memoria

sarebbe troppo pesante per lo M.U, semplicemente può succedere che quella risoluzione non sia al momento nella memoria, allora il Sistema Operativo è lui che ricarica la parte mancante e fa riportare a tutto va bene.
Se alla prima trovata della M.U, perché non trova nella cache che ha a bordo la risoluzione, arriverà al Sistema Operativo e dice "nah! Questo è un indirizzo che non ho neanche nella tabella quella estesa che gestisco io" a quel punto il Kernel genera il segnale. Comunque parte da una operazione della M.U. Divisione per 0 o istruzione illegale vengono generati direttamente dal processore.

pausso (min 1:07:45)