IA-32 Přepínání úloh

Obsah

IA – 32 Přepínání úloh

a) Účel, princip funkce, logická struktura

Účel

- Zvýšení výkonu procesoru. Efektivněji využívá procesor pomocí odstranění prostojů. Pokud úloha z nějakého důvodu čeká (např. není přísun dat), pracuje mezitím úloha jiná
- Používá princip pseudoparalelismu - výpočetní systém vypadá opticky lépe

Princip funkce

- Chráněný režim umožňuje, aby bylo zavedeno do paměti více úkolů najednou, přičemž aktivní je pouze jeden z nich
- **Předávání** řízení mezi jednotlivými úkoly **hardwarově zajišťuje mechanismus přepínání úloh** (hardwarově kvůli rychlosti)
- Přepnutí úlohy předpokládá uložení kontextu pro aktuální proces a nahrání jiného kontextu pro jiný proces.

Ukládání a nahrávání kontextu je realizováno atomicky (stejně jako instrukce)

Časově závislé chyby

- Mezi procesy musí existovat komunikace a synchronizace, aby nedošlo k časově závislým chybám v důsledku dynamického přidělování zdrojů v kritických sekcích (v jeden moment soupeří více procesů o jeden systémový zdroj)
    - **deadlock** – mrtvý bod (každý proces má nějaké zdroje ale ne všechny potřebné)
    - **starvation** – vyhladovění (více procesů provádí aktivní čekání a neuvolní své zdroje – znemožní práci ostatním procesům)
- **Kritická sekce** je kus kódu, který nesmí být rozložen a musí být proveden najednou

Řešení časově závislých chyb

Zákaz přerušení

- možné pouze pro jednouživatelské OS
- dlouhodobý zákaz může vést k tomu, že procesor některé přerušení propásne (fatální důsledky)

Aktivní čekání

- cyklicky se testuje proměnná (zámek)
- vstoupí-li proces do kritické sekce, nastaví proměnnou na true a po výstupu na false
- kontrola stavu proměnné zabírá procesorový čas
- pokud proces při činnosti zkolabuje, zámek nikdo nenastaví na false (fatální)

Instrukce TSL (Test Set and Lock)

- hardwarová realizace aktivního čekání
- zajistí čtení i zápis do buňky paměti nepřerušitelným způsobem

Synchronizační prostředky OS

- **semafor** – objekt obsahuje celočíselný čítač a frontu procesů
- **monitor –** objekt, který má definované proměnné a procedury (pokud je v monitoru proces, další proces nemůže monitor zavolat)
- **předávání zpráv –** procesy nepotřebují společnou paměť. Při asynchronním módu proces pošle zprávu a pokračuje v činnosti. Při synchronním módu pošle zprávu a čeká na odpověď.

Řízení procesů

Rozvrhovač (Scheduler) – proces jádra OS, který rozhoduje o přidělování systémových zdrojů procesům

- **Dobrovolná výměna (nonpreemtive)** – proces pracuje tak dlouho, dokud se sám neodstaví (vhodné pro databázové systémy s jedním řídícím procesem)
- **Nucená výměna (preemtive) –** na konci přiděleného časového intervalu je procesor procesu odebrán. Důležitý je poměr kvanta času a doby odezvy (čím více času proces dostane, tím pomalejší je odezva)
    - **Cyklická obsluha** – každý proces má stejné kvantum času
    - **Podle priority** – proces má podle kritéria přidělenou prioritu (fixní nebo dynamickou) a z množiny připravených procesů se spustí ten s nejvyšší prioritou.
    - **Fronty s prioritou** – více front s prioritou. Fronta přidělí kvantum času a každá další fronta kvantum dvakrát větší než předchozí. Proces startuje s nejvyšší prioritou a po odstavení se zařadí na konec fronty s nižší prioritou. To vede k tomu, že **krátký proces** bude **rychle vykonán** a **dlouhý méněkrát přerušen**.

Kritéria podle druhu OS

- **Systému dávkového zpracování**
    - propustnost (počet operací za jednotku času)
    - doba obrátky (doba potřebná k vykonání úlohy a stupeň využití CPU)
- **Interaktivní systémy**
    - Doba odezvy a proporcionalita mezi procesy (uspokojení uživatelů rychlostí OS)
- **Systémy reálného času**
    - Nízká doba odezvy (krátké procesy)
    - Zabránění degradací poskytovaných služeb

Kontext

Kontext je soubor prostředků a zdrojů, nutných pro obnovení pozastaveného procesu. Jde o registry procesoru, odkazy na zásobníky jednotlivých úrovní oprávnění, zpětný ukazatel, selektor LDT (local description table) a odkaz na mapu portů.

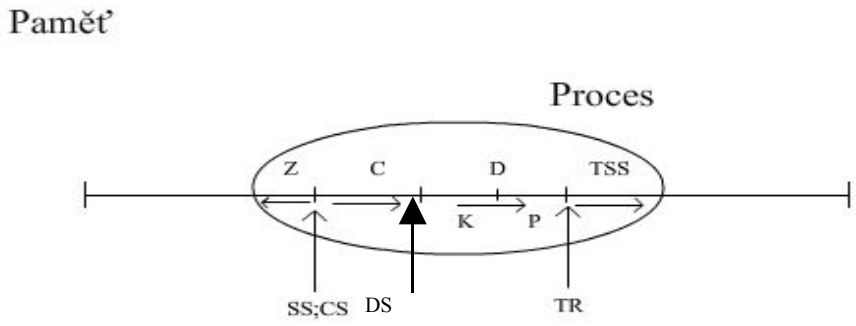
Typy úloh

1. program
2. služba OS
3. obsluha přerušení
4. obsluha výjimečné situace

Struktura úlohy v paměti

Logické části

- každá **úloha je složena ze 2 logických částí**
  1. výkonný prostor (segmenty kódu, dat, zásobníku)
  2. segment stavu úlohy (TSS – Task State Segment)
- každá úloha je jednoznačně identifikována ukazatelem na svou TSS



Z – segment zásobníku (na začátek segmentu odkazuje registr SS)
C – segment kódu (na začátek segmentu odkazuje registr CS)
D – segment dat obsahuje blok konstant (K) a proměnných (P) a odkazuje na něj registr DS
TSS – segment stavu úlohy (odkazuje na něj registr **TR – Task Register**)

TSS – popis funkce

TSS je struktura, sloužící k uložení stavu (kontextu) procesu v momentě přepnutí úlohy. Na umístění TSS odkazuje registr TR (task register). TSS je **systémový paměťový objekt**.

TSS – logické dělení dat do skupin

1. zpětný ukazatel (selektor přerušeného procesu)
2. ukazatele zásobníků úrovní oprávnění 2 až 0 (SS,SP)
3. registry procesoru
4. selektor LDT (selektor položky GDT, která specifikuje LDT procesu)
5. odkaz na mapu portů

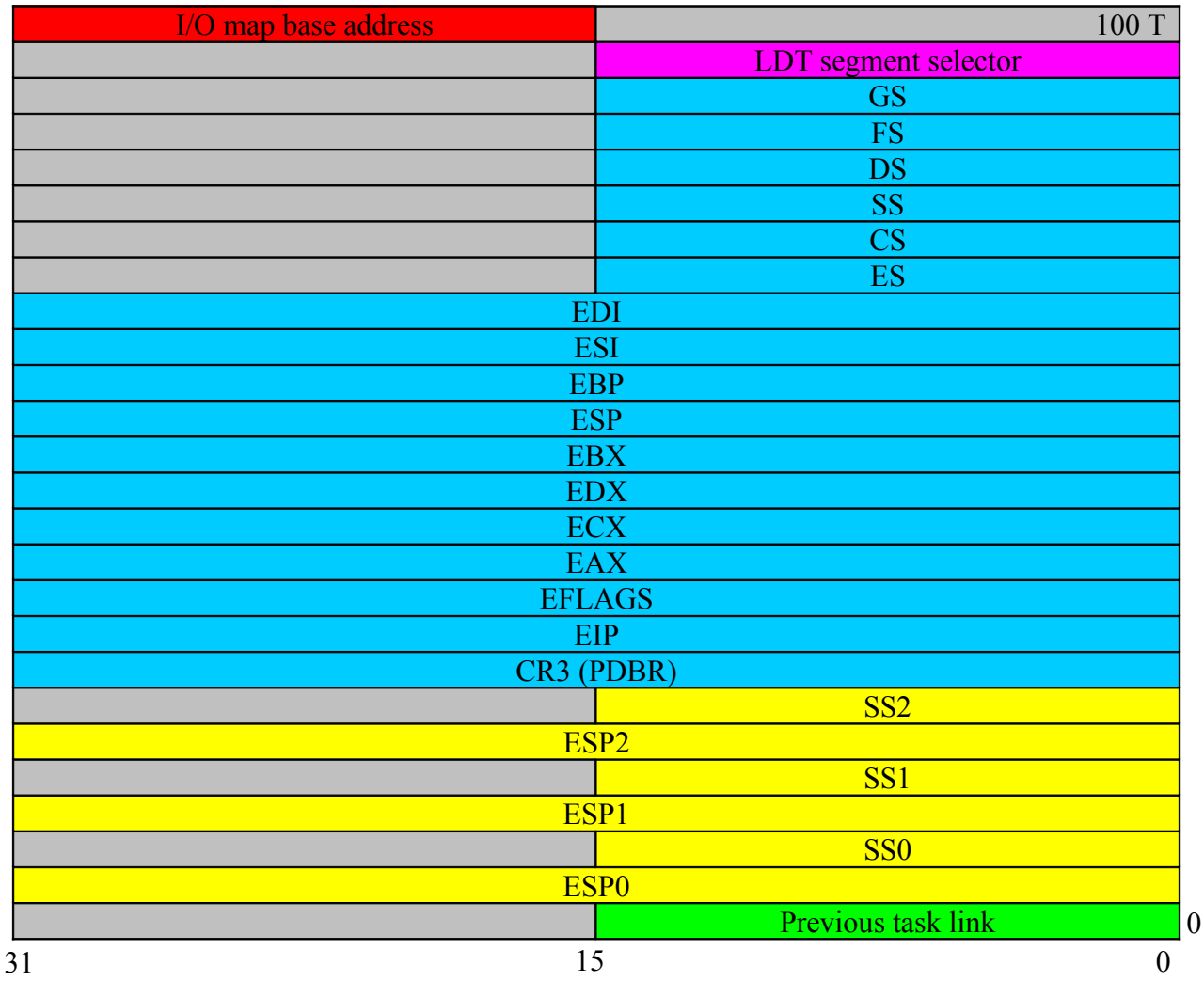
TSS – rozdělení dat

1. statická – nastaví se při inicializaci procesu (např. odkaz na mapu portů nebo ukazatele zásobníků)
2. dynamická – změní se při přepnutí úlohy (např. registry procesoru)

TSS - velikost

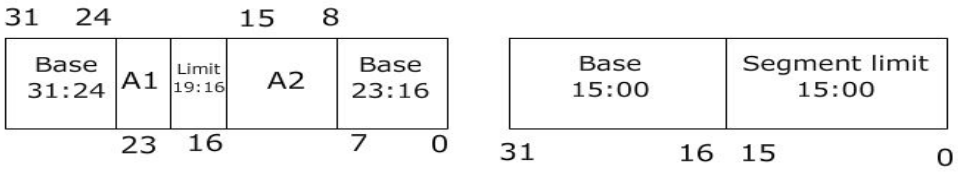
1. pro 32 bitové prostředí minimálně 104 bajty
2. pro 16. bitové prostředí 42 bajtů

TSS – struktura



- odkaz na mapu portů
- selektor LDT
- registry procesoru
- ukazatele zásobníků jednotlivých úrovní oprávnění
- zpětný ukazatel
- rezervované bity nastavené na 0

TSS – formát popisovače



Base – Bázová adresa segmentu (celkem 32 bitů)
Limit – Velikost segmentu (celkem 20 bitů)
A1, A2 – pole atributů (celkem 12 bitů)

b)Způsoby předání řízení, systémové objekty, režim V86

Přepnutí úlohy

Je realizováno atomicky a hardwarově. Kontext stávající úlohy se uloží pomocí TSS a kontext nové úlohy se nahraje do registrů.

Přímé přepnutí

Přímé přepnutí se realizuje, když se kontext přepíná do segmentu se stejným oprávněním a stejným typem kódu (16. nebo 32. bitovým). Děje se tak instrukcí (CALL, volání; FAR JMP, skok). Bezprostředně aktivuje objekt TSS (přímý odkaz na TSS), jehož popisovač musí být uložen v GDT (Global Description Table).

Nepřímé (chráněné) přepnutí

Používá se při přechodu do segmentu s vyšším oprávněním, při přerušení a při přechodu na jiný typ kódu (16 => 32 nebo obráceně). Předpokládá použití brány (**TG – Task Gate** – systémový objekt). **Popisovač brány** (TG) může být uložen **v GDT** (global description table) , **LDT** (local description table), **IDT** (interrupt description table). Důvodem **použití brány** je umožnit úlohám **selektivní přístup k** určitým **procesům** (kontrolují přepínání procesů – „chrání“)

Popis brány

Účel

1. předání řízení do segmentu s vyšším oprávněním (**Call gate**)
2. předání řízení pro nemaskovatelné přerušení (**Trap gate**)
3. předání řízení pro maskovatelné přerušení (**Interrupt gate**)
4. zpřístupnění TSS (**Task gate**) – umožňuje nepřímé předání řízení
5. používá se při přechodu mezi 16. bitovým a 32. bitovým kódem

Formát popisovače TG (Task gate)

- popisovač má klasický formát (8 Bytů), ale významné jsou jen 3 Byty
- spodní 2 byty báze jsou ukazatel na popisovač TSS úlohy a bity spodního bytu atributů udávají třídu objektu (TG = 5), přístupová práva a platnost položky

Popisovač brány

1. specifikuje platnost popisovače
2. definuje vstupní bod procesu, kterému bude předáno řízení (selektor segmentu kódu a offset)
3. určuje prioritu procesu, počet předaných parametrů zásobníků (dochází-li k přepnutí zásobníků, určuje počet a šířku parametrů (slov), které budou mezi zásobníky kopírovány(např. při přerušení se žádné parametry nepředávají – Trap, Interrupt gate))

Režim V86

Účel

Odstraňuje některé chyby, upravuje přerušení a hlavně v rámci chráněného módu **umožňuje spouštět programy určené pro procesor 8086 bez nutnosti rekompilace** (změn v kódu).

Princip

Režim V86 umožňuje provozovat 1 nebo více procesorů 8086 na 1 procesoru 80386 nebo vyšším. Jde o softwarovou emulaci.

Funkce

Je to typ chráněného režimu. V86 poskytuje podporu pro spouštění aplikací napsaných pro 8086 (16. bitové aplikace) – Reálně vypadá jako konzole (Dosovské okno). Program má k dispozici 1 MB paměti (jakoby svůj virtuální prostor). Ochranné mechanismy do spuštěného procesu 8086 nezasahují, ale hlídají akce, které mohou ovlivnit jiné probíhající akce. Pokud se aplikace snaží dostat mimo rámec virtuálního stroje, vznikne přerušení, které je obslouženo v chráněném módu.
Přístup k zařízením vstupu a výstupu (IN/OUT) je omezen mapou povolených portů. **Mapa portů** má 65 535 portů pro 8. bitovou velikost každého portu a 32 767 pro 16. bitovou velikost každého portu. Každý port má v mapě portů přiřazen jeden bit, který říká, jestli je port povolen (Pokud má hodnotu 1, je zakázán. Pro hodnotu 0 je povolen). Pokud se aplikace snaží přistupovat k portům přímo, vznikne přerušení, které je obslouženo přepnutím do chráněného módu podle příslušného popisovače (tzn., že je aplikace napsána špatně).

c) Aplikace, operační systém a správa procesů

Windows XP je operační systém **víceúlohový** a **preemptivní**. Znamená to, že každý spuštěný proces má striktně danou dobu, po kterou může být vykonáván. Po uplynutí této doby je mu odebrán procesor. Výjimkou je **činnost jádra**, která **nemůže být přerušena nikdy (non preemtive)**.

Základní struktura (vrstvy a moduly)

- hardware
- HAL vrstva
- Jádro (kernel)
- Exekutiva
- Volací rozhraní kernelu
- API Win 32
- Servery prostředí a subsystémy zabezpečení

Správce procesů

Zabezpečuje služby pro vznik, použití a zrušení procesů, vláken a úloh a nalézá se v exekutivě OS.

Vlákno

- Základní běhová jednotka (Nejmenší spustitelný kód). Vlákna lze sdružovat do procesů

Proces

- Objekt, který reprezentuje instanci vykonávajícího se programu
- Správní jednotka. Nedělá nic, pouze poskytuje kontext pro spouštění vláken. Je to virtuální adresový prostor, kód, data a přidělené zdroje.

Úloha

- Organizační jednotka, ve které je sdruženo více procesů
- Slouží k lepšímu využití zdrojů
- Každé úloze lze přiřadit afinitu (ke kterému procesu tíhne- kde má přidělené zdroje…)

Synchronizace

Synchronizace mezi procesy slouží k tomu, aby nedošlo ke korupci (zkomolení) dat. Proto jsou zásadní datové struktury (procesy) spravovány jako systémové objekty. Manažeři jsou dvojího druhu:

- Signalizační (signalizují událost, reagovat může více vláken)
- Synchronizační (reagovat může jen jedno vlákno)

Synchronizace ve víceprocesorovém prostředí zajišťuje mechanismus **SPIN LOCK**. Zajišťuje vzájemně se vylučující přístup vláken, běžících na různých procesorech ke sdíleným datům. Je-li SPIN LOCK nastaven, nedojde k přepnutí kontextu, dokud svou činnost jádro neukončí. Vlákna na ostatních procesorech můžou dál pracovat.

Objekty

Objekt je netransparentní datová struktura (data + množina operací). Manažer objektů manipuluje s objekty. Metody pro vytváření, rušení, přidávání nových typů objektů. Umožňuje modulům pro instance definovat bezpečnostní mechanismy.
Objekty lze rozlišit podle jména nebo systémového identifikátoru (HANDLE). Pro daný proces je handle unikátní.
Při manipulaci s objekty existují virtuální funkce (Create, Open, Close, Delete…)

Typy manažerů objektů

Manažer objektu EVENT

- Slouží k řízení vláken. Podle detekovaných událostí synchronizuje akce

Manažer objektu Mutant a Mutex

- Řídí bezkonfliktní přístup ke sdíleným datům. V daný moment zpřístupní data pouze jednomu vláknu, ostatní vlákna požadující sdílená data jsou pozastavena. (Mutant pracuje v režimu jádra i uživatele a Mutex pouze v režimu jádra)

Manažer objektu Semaphore

- Objekt soužící ke správě zdrojů přidělených vláknům. Čítač, který zajišťuje souběžný, ale vzájemně se vylučující přístup k chráněným prostředkům.

Manažer objektu Timer

- Odměřuje časové intervaly
- Plánuje periodicky se opakující děje
- Ukončuje akce, které vyčerpaly plánovaný čas

Vznik procesu a vlákna

1. Iniciátor pošle zprávu Win 32 API, že hodlá vytvořit proces
2. Procedura CreateProceses() v původním procesu předá řízení manažeru procesů
3. manažer procesů zavolá manažer objektů a ten vytvoří objekt procesu a předá příslušný HANDLE Win 32 API
4. Win 32 API znovu zavolá manažer procesů aby vytvořil vlákno pro proces a vrátil HANDLE vláknu

Stavy vlákna

2. Připraven
    - Vlákno může být spuštěno
3. V pohotovosti
    - Vlákno s nejvyšší prioritou nastaveno do stavu pohotovosti
4. Běží
    - Přidělení procesoru se uvede do stavu „běží“
5. Čeká
    - Čeká na přidělení zdrojů
6. Přechodný stav (pozastaveno)
    - Uplynutím časového intervalu, spuštěním vlákna s vyšší prioritou, blokovacím signálem dispečera procesů (např. čekání operace přenosu dat)
7. Ukončen
    - Slouží k uvolnění přidělených zdrojů
    - Dispečer jádra vychází z prioritního schématu o 32 úrovních
    - Každé vlákno má tabulku stavu (priorita, afinita, využití kvanta času…)

Příklad přepnutí úlohy ve Win XP

Konkrétní aplikace (ne pro Windows XP) běží v systému Windows XP pomocí serverů prostředí. Při přepnutí úlohy (softwarová činnost) musí server prostředí předat příkaz Win 32 API. Ta přes exekutivu a jádro pošle požadavek vrstvě HAL, která převede požadavek na příkazy konkrétního stroje. Pak teprve může být přepnutí úlohy podpořeno hardwarem.