IA 32

Základním rysem IA-32 je hardwarová podpora funkcí operačního systému.

Hardwarová podpora funkcí OS: procesor napomáhá OS v efektivní a spolehlivé realizaci příslušné funkce

(realizuje elementární úkony na obvodové úrovni)

To předpokládá bezchybnou spolupráci procesoru a operačního systému (kooperace HW a SW).

funkce OS:správa procesů: více úloh, více uživatelů

správa prostředků více procesorů

správa paměti: ochrana, stránkování, cachování

odpovídající obsluha přerušení a výjimečných situací - chybové stavy, ladění

Správa paměti

**Účel:** přidělování a ochrana paměti, evidence prostoru, realizace virtuální paměti

**Fyzická paměť z pohledu:** paměťové jednotky - lineární prostor (flat model)

procesoru - rozdělena na bloky (segmenty)

rozdělení – obstarává procesor (není samoúčelné)

**Segmentace -** metoda přidělování paměti založená na rozdělení logického adresového prostoru na bloky (segmenty)

segment - souvislý paměťový prostor o délce až 64 KB určený parametry

parametry: báze segmentu - adresa začátku

limit segmentu - délku ve slabikách

přístupové práva - 4 úrovně oprávnění (umožňují vyjádřit množství „důvěry“ poskytnuté procesoru)

typy segmentu: pouze vykonat

pouze číst

vykonat a číst

číst i psát

paměťové bloky - úseky fyzického adresového prostoru s obecně rozdílnou velikostí, do nichž jsou umístěny segmenty

Adresový prostor - samostatný způsob číslování paměťových buněk

segmentové registry (16 bitových): CS, DS, SS, ES, FS, GS

segmentační jednotka – převádí logické adresy na lineární (případně na fyzické)

Logická adresa (48 bitů) - selektor segmentu (16 bitů) a offset (32 bitů)

Lineární adresa (32bit. číslo) - předána segmentační jednotkou jednotce stránkovací

Fyzická adresa - binární číslo použité na adresových linkách rozhraní CPU (paměť)

složená z: segmentu a ofsetu

získána jako - součet binárního čísla a offsetu

binární číslo: udává začátek bloku v paměti stroje

Hodnota ofsetu – vzdálenost od začátku bloku paměti (definován segmentovou částí adresy)

mechanizmy převodu segmentové části adresy na binární číslo:

reálný režim - hodnota segmentového registru se násobí šestnácti (posune o čtyři binární řády)

chráněný režim - hodnota segmentového registru je jen ukazatel do tabulky popisovačů objektů (deskriptorů)

# **zpětnou kompatibilitu zajišťuje -** **virtuální režim 8086** (v rámci chráněného módu)

Z pohledu uživatele režim V86 znamená provozovat jeden nebo i více procesorů 8086 na jednom procesoru 80386 (nebo vyšším). Ochranné mechanizmy chráněného režimu do probíhajícího procesu 8086 nezasahují, hlídají však akce, které mohou ovlivnit ostatní probíhající procesy. Přístup na zařízení V/V je omezeno mapou povolených portů. Je-li příslušný bit nastaven, není přístup povolen. Přerušení je obsluhováno přepnutím do chráněného režimu podle příslušného popisovače.

**Tabulka popisovačů objektů -** v ní uložena počáteční adresa bloku v paměti (x bajtů)

obsahuje - jednotlivé položky

**položka -** dlouhá 8 bajtů

obsahuje -4 pole různé délky

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 63 | báze (bity 24-31) | atributy (vlastnosti) 12 bitů | báze (bity 0-23) | limit 20 bitů | 0 |

bity pole atributů mají následující význam:

15 G – Granulita – limit je v bajtech nebo stránky 4kB

14 Db – default 16/32

13 O

12 Av1 – volný bit pro použití návrháři systému

11-8 Li (9-6)

7 P – objekt existuje v paměti

6-5 DPL – přístupová práva popisovače

4 DT – typ popisovače systém/segment DT = 1 - systém

3 CD – typ segmentu kód/data CD = 1 - kód

2 EC – interpretuje se podle toho, jedná-li se o kód nebo data

1 WR ----------------------------------||-----------------------------------

0 A – indikuje přístup (použitím popisovače). Používá systém k sledování aktivity

typy: **globální (GDT) -** unikátní a mohou v ní být umístěny objekty libovolné třídy mající všeobecnou platnost

systémového registr GDTR - udává počátek této tabulky a její limit v paměti

realizován jako - 48 bitový registr

obsahuje: bázi (32 bitů)

limit (16 bitů) tabulky

#### lokální (LDT)- může být vícero a mohou obsahovat pouze objekty mající místní (task) platnost

systémový registr LDTR - pro výběr aktivní tabulky

udává částí (svou): viditelnou - 16bitový ukazatel (selektor) do GDT

neviditelnou - počátek a limit

#### pro přerušení (IDT) - unikátní a může obsahovat pouze objekty třídy systém obsluhující přerušení.

systémový registr IDTR – udává začátek tabulky a její limit

realizován jako - 48 bitový registr

obsahuje: bázi (32 bitů) a limit (16 bitů) tabulky

formát ukazatele (selektoru)

segmentový registr má šířku 16 bitů (skutečná šířka 64 bitů):

bity 0 a 1 – úroveň oprávnění (RPL requester´s prvilege level)

bit 2 – volba tabulky popisovačů (TI table indicator) - globální (=0) nebo lokální (=1)

bity 3 - 15 – index do tabulky popisovačů objektů

index - ukazuje na konkrétní položku v tabulce

objekty umístěné v operační paměti dělíme do tříd:systém a segment

Třída systém obsahuje objekty: Nepovolená hodnota

TSS aktivního a neaktivního procesu

Brána pro předání řízení

Brána zpřístupňující TSS

Brána pro maskovatelné a nemaskovatelné přerušení

LDT

brána- systémový objekt uložený v tabulce popisovačů

účel: zabezpečuje nepřímé, chráněné předání řízení mezi procesy: s různou prioritou

realizovanými v různém kódu (16 a 32 bit)

používá se k předání řízení při přerušení (nepředávají se žádné parametry - trap a interrupt gate)

funkce:předání řízení do segmentu s vyšším oprávněním (Call Gate)

předání řízení pro nemaskovatelné přerušení (Trap Gate)

předání řízení při maskovatelném přerušení (Interrupt Gate)

zpřístupnění segmentu stavu procesu (Task Gate)

# **Popisovač brány**

# **funkce:** specifikuje platnost popisovače

definuje vstupní bod procesu (selektor segmentu kódu a ofset), kterému bude předáno řízení

určuje prioritu tohoto procesu

při přepnutí zásobníků určuje kolik parametrů (slov) a jaké šířky bude mezi zásobníky kopírováno

(určuje typ brány a D bit popisovače CS volajícího procesu)

K přepnutí dojde – při předává řízení nepřizpůsobivému procesu s vyšší prioritou. Přepíná se na nový, dočasný zásobník!

důvod: snaha procesy bezezbytku odizolovat (zabránit havárii procesu a eliminovat přístup procesu s nižší prioritou)

snaha zabránit havárii procesu s vyšší prioritou v důsledku podtečení zásobníku a také eliminovat jakýkoli úmyslný či neúmyslný přístup procesu s nižší prioritou k datům procesu s vyšší prioritou (přes sdílený zásobník).

## **TSS (Task State Segment)** – struktura sloužící k uložení stavu (kontextu) procesu v momentě přepnutí

data zde uložená dělíme na: statické - nastavují se při inicializaci procesu

dynamické - mění se při přepnutí

logické členění: zpětný ukazatel (selektor přerušeného procesu)

ukazatele zásobníků úrovní 2 až 0 (SS, SP)

registry procesoru

selektor LDT (selektor položky GDT specifikuje LDT procesu)

# mechanismus správy paměti: spuštěn když dojde ke změně obsahu libovolného segmentového registru

Vložená hodnota je interpretována jako ukazatel (selektor) do tabulky popisovačů objektů.

Z příslušného registru (GDTR, IDTR, LDTR) zjistí procesor, kde je tabulka v paměti uložena (bázi a limit).

Procesor zkontroluje bit 7 prvého bajtu atributů. Je-li nastaven (=1) je položka platná, ne-li dojde k přerušení (11 nebo 12).

Procesor zkontroluje bit 4 prvého bajtu atributů a určí, o jaký typ objektu se jedná.

Další postup závisí na typu objektu: Jedná-li se o systém, pak spodní 4 bity prvého bajtu atributů určují konkrétně, o jaký druh systémového objektu se jedná (TSS, brána). V případě segmentu pak tyto bity určují způsob manipulace s objektem (pouze vykonat, číst a vykonat, pouze číst, číst i psát).

Zkontrolují se přístupová práva – bity 0 a 1 selektoru a bity 5 a 6 prvého bajtu atributů.

V případě selhání je generováno příslušné přerušení (13 – general protection).

Položka z tabulky popisovačů je zanesena do neviditelné části segmentového registru.

# Stránkování

Dovoluje programům užívat virtuální paměť, která je větší než skutečná fyzicky přítomná operační paměť.

fyzická paměť - rozdělena na bloky konstantní délky (rámce)

virtuální paměť - rozdělena na stránky délky 4kB (či 4MB, 2MB), které jsou uloženy na externím zařízení

velikost stránky se ovládá: bity 4 a 5 v registru CR4 (bity PSE a PAE)

2 MB - nastavením bitu 7 (PS) v položce adresáře stránek

Pokud se používají stránky o velikosti 2MB je uvedený překlad pouze dvoustupňový (2 MB = 21 bitů).

virtuální prostor - vždy omezen na 4 GB (32-bitová adresa)

Logická adresa 32 bitů – číslo stránky (20 bitů) a adresy ve stránce (offset, 12 bitů)

číslo stránky - index do tabulky stránek, obsahující bázové adresy rámce přidělené stránce, do které patří logická adresa

stránkování: stará se o přepočet adres (manipulace se stránkami je záležitost os)

lze jej vypnout a zapnout (ovládá to os)

**mechanizmus stránkovaní paměti** (pracuje nad mechanizmem segmentace)

Procesor poskytuje tomuto procesu potřebnou hardwarovou podporu (stránkovací jednotka procesoru).

Transformace virtuální adresy na fyzickou se provádí při každém přístupu do paměti. Způsobů provedení transformace existuje více, vždy se jedná o časově náročnou operaci. Procesor má proto implementovanou vyrovnávací paměť (asociativního typu) pro posledně transformované adresy (TLB – Translation Lokk-aside Buffer).

pro testování a ladění mechanizmu: 32bitové registry TR6 a TR7

Stránkovací jednotka: transformuje 32 bit. lineární (virtuální) adresu na fyzickou tak aby byla splněna podmínka

signalizuje stav při nepřítomnosti potřebné stránky v paměti (přerušení #PF – page fault)

Podmínka - výsledná adresa není větší než limitní hodnota systému reálně přítomné operační paměti

registry stránkovací jednotky procesoru: CR3 – obsahuje fyzickou adresu adresář stránek

CR2 – v tomto registru je v případě výpadku stránky její lineární adresa

CR0 – v tomto registru se zapíná stránkování nastavením bitu 31

Princip činnosti: V případě zaplnění paměti je uvolněn některý rámec podle určitého kritéria (nejmenší počet přístupu k obsahu rámce za časovou jednotku). Jeho obsah je uložen jako stránka do odkládacího souboru na vnější paměťové médium. Získané volné místo je přiděleno jinému procesu, neboli stejné místo v paměti je sdíleno, různými procesy v daný časový moment (no vlastně pouze jedním procesem a ostatní jsou v očekávání na vnějším paměťovém médiu). Aby programem potřebná stránka byla následně natažena z externího média do rámce v operační paměti zajišťuje operační OS.

průběh přepočetu adresy: Horních 10 bitů se použije jako ukazatel do adresáře stránek. Počáteční adresa adresáře stránek je uložena v registru CR3 (musí být vždy dělitelná 4k). Adresář stránek je struktura 1024 čtyřbajtových položek. Spodních 12 bitů jsou atributy. Bit 0 určuje platnost položky. Je-li nenastaven pak požadovaná stránka s tabulkou stránek není v paměti. Vznikne přerušení (#PF) a rutina obsluhy se postará o natažení stránky do paměti.

Tabulka stránek má 1024 čtyřbajtových položek. Horních 10 bitů je finální adresa a spodních 12 atributy. Bit 0 opět určuje platnost položky. Neplatnost znamená, že stránka není v paměti a horních 31 bitů položky určuje místo uložení stránky na vnějším nosiči (většinou). Přepočet adresy vyžaduje 2x číst z paměti – zrychluje TLB

S příchodem Pentia Pro byla adresa rozšířena na 36 bitů. Což znamená, že v systému může být až 64 GB fyzické paměti a 4 GB virtuální paměti je vždy lineárně mapováno do spodních 4 GB reálné paměti. Uvedený problém řeší režim (mód) PAE-36. Používá se tříúrovňová transformace lineární adresy. Zapnutí tohoto režimu se provádí nastavením bitu 5 PAE (physical address extension) v registru CR4. Nejvyšší dva bity 32-bitové lineární adresy ukazují do tabulky (o čtyřech položkách) adresářů stránek. Každý adresář stránek může obsahovat až 512 položek o délce 64 bity. Ostatní je v zásadě stejné, jako v případě dvouúrovňového překladu.

# 

# systém ochran

Procesoru spuštěnému v chráněném režimu je v závislosti na stupni důvěry přidělena určitá úroveň oprávnění.

účel: zajistit, že každý proces má přístup jen ke své části operační paměti a nemůže (záměrně, ani omylem) číst, nebo dokonce zapisovat a poškodit paměť jiného procesu

Každá úloha je sestavena z kódu a dat, které jsou uloženy v příslušném segmentu. Každému segmentu je přiřazena v závislosti na jeho obsahu jistá úroveň oprávnění uvedená v popisovači příslušného segmentu.

úrovně oprávnění (čím vyšší číselná hodnota tím menší oprávnění):

0 – jádro OS – řízení procesoru

1 – služby OS – přidělování prostředků, plánování úloh

2 – systémové programy z knihoven, souborový systém

3 – uživatelské aplikace

indikátory oprávnění:

DPL – přístupová práva popisovače segmentu (Descriptor Privilege Level)

CPL – úroveň oprávnění procesu – selektor CS (Current Privilege Level)

RPL – požadovaná přístupová práva – selektor segmentu (Requested Privilege Level)

EPL – efektivní přístupová práva – numerické maximum CPL a RPL

úrovně oprávněnídefinovány v: segmentovém registru kódu – CPL

selektoru popisovače – RPL

deskriptoru objektu – DPL

kategorie kontrol činnosti (v chráněném módu): limit objektu

typ objektu

úroveň oprávnění

omezení pro adresovatelné oblasti

omezení pro vstupní body procedur

omezení v použití instrukčního souboru

způsoby ochrany bloků operační paměti:

metoda zámků a klíčů (poprvé použita v Systému 360 Ibm) :

#### rozdělení paměti na bloky: Paměť byla rozdělena na bloky o délce 2kB a každému bloku byl přidělen čtyřbitový ochranný kód (klíč). V registru stavového slova programu (PSW) byla čtyřbitová kombinace (zámek). Pokud se program pokoušel přistupovat k bloku, jehož klíč neodpovídal zámku, byl hardwarově zablokován.

#### použití speciálních registrů (báze, limit): Při inicializaci programu (procesu) se do registru báze zanese počáteční adresa přiděleného adresového prostranství a do registru limit délka (velikosti) prostranství.

výhodou - použití báze řeší i problém relokace adres – finální adresa se získá jakou součet relativní adresy (od začátku bloku, offset) a báze.

nevýhodou - nutnost provést sčítání a porovnání při každém přístupu k operační paměti (časově náročné)

### metody evidence použití operační paměti:

#### metoda bitové mapy: rozdělení pamětí na bloky konstantní délky – alokační jednotky. S každou jednotkou je spojen jeden bit v bitové mapě. Nastavení bitu signalizuje, že blok je použit (=0 -> volný).

velikosti alokační jednotky: bajty - bitová mapa veliká (spotřebuje hodně paměti)

kB - poměrně malá (ale přidělují se často bloky zbytečně velké – plýtvání)

nevýhoda - poměrně složitý postup při vyhledávání volné bloku paměti určité velikosti

#### spojový seznam (zřetězený): seznam jehož datové položky jsou v paměti rozptýleny,avšak každá nese informaci, kde je následující položka

třídění seznamu podle: adresy - výhodné při uvolnění bloku

velikosti bloku - výhodné při přidělování podle požadavků

##### každá položka obsahuje pole: charakteristika bloku

délka bloku

adresa následující položky seznamu

Uvolnění bloku realizuje složitější algoritmus – každý blok má dva sousedy (s výjimkou prvého a posledního)

varianty: obsazeno na obou stranách - díra

volno vlevo

volno vpravo

volno na obou stranách - vznikne jeden souvislý blok místo tří původních

paměti cache (většinou se používá dvou a více úrovňový systém)

cache – rychlá, malá paměť vložená mezi procesor a hlavní paměť

účel: zkrátit efektivní vybavovací dobu hlavní paměti.

délka bloku byla 16 bajtů (486), 32 bajtů (pentium P6), 128 bajtů (pentium 4) a 64 bajtu (pentium M)

L1 cache: umístěna přímo v procesoru

rozdělena na datovou a instrukční

(split) dělená na: kód - 4cestná asociativní paměť

data- 2cestná asociativní paměť

L2 cache: tvoří jí samostatné obvody umístěné na základové desce

(unified) společná - 8cestná asociativní paměť

Do Pentia Pro se jedná o blokující paměť (po stavu MISS se zablokuje) následné procesory už mají neblokující se (má zásadní význam při zpracování instrukcí mimo pořadí (out-of-order)).

úspěšné použití paměti předpokládá vyřešit:

problém mapování - přiřazení bloků hlavní paměti blokům cache

problém aktualizace - vhodný algoritmus výměny (vytěsňování) dat z cache

problém konzistence - zabezpečení shody obsahu (zvláště při více procesorech)

## organizace paměti cache (mapování):

### plně asociativní mapování (nejlepší a nejdražší) – vhodné pouze pro nevelké systémy

### Llibovolný blok hlavní paměti může být umístěn v libovolném bloku paměti cache. Porovnání adres bloků v paměti cache s požadovanou adresou je plně asociativní. Plně asociativní mapování se používá pouze pro TLB (stránkování).

Prověření, zda se položka nachází v paměti cache předpokládá porovnat najednou adresu se všemi tagy uloženými v cache (porovnání po bitech). Což je obvodově složité (-> nákladné a pomalé).

### přímé mapování (nejjednodušší) - určité bloky hlavní paměti jsou přiřazeny právě jednomu bloku paměti cache. Do velikosti cache cca 900 bajtů nejefektivnější organizace (není třeba vypočítávat, který blok bude vytěsněn)

nevýhoda - pravděpodobnost úspěšného nalezení položky (HIT) v cache je nižší.

### částečně asociativní paměť - kombinace uvedených metod. Dnes nejrozšířenější organizace pamětí cache (několikachodé paměti cache). Je třeba provést n porovnání k získání výsledku.

**mechanizmy uvolňování dat z paměti cache (aktualizace):**

**LRU (Least Recently Used) -**  metoda výměny založená na předpokladu, že dlouho nepoužitá stránka bude ještě dlouho nepotřebná a proto je možné ji odstranit. Realizace algoritmu je nákladná na zdroje. Předpokládá setříděný svázaný seznam (frontu) všech stránek v paměti přítomných. V čele fronty je právě použitá a v týlu nejdéle nepoužitá stránka. Při každém přístupu do paměti je třeba frontu aktualizovat.

**FIFO -** principem je fronta – nejstarší stránka je první a přidává se na konec. Vyměňuje se první.

**zajištění konzistentnosti dat** (největší problém)

Strategie přesouvání dat mezi těmito paměťmi je založena na skutečnosti, že přístupy do operační paměti mají jistou časovou a místní lokalitu. Přístup k datovým položkám ale vykazuje poněkud odlišné závislosti než přístup k instrukcím.

**účel:** dosáhnout toho, aby se každá změna co nejdříve objevila v hlavní paměti

**strategie:** současného zápisu (write through) - data společně do obou pamětí (nejjednodušší, ale pomalé)

Modifikuje data současně jak v cache tak v hlavní paměti. Hlavní paměť je však pomalá a brzdí cache (řešení: uložení zapisovaného do speciálních vyrovnávacích registrů). Stejně, ale po jistý čas je nesoulad mezi daty v hlavní paměti a cache. Řeší se to různě, podstatou ne nastavení příznaku neplatnosti dat.

Nevýhoda - zvýšené nároky na šířku pásma paměťové sběrnice.

zpožděný zápis (write back) - zapisovat (uklízet) změny do hlavní paměti při vyřazování bloku

ukládá změny pouze do bloku v paměti cache. Blok je přitom označen jako modifikovaný (dirty). V hlavní paměti se změny projeví až při uvolnění bloku z cache (úklidu). Uklízejí se pouze modifikované bloky! Tato strategie snižuje komunikaci mezi hlavní pamětí a cache, ale vyžaduje složité mechanizmy pro zajištění platnosti dat mezi pamětmi různé hierarchie (nevýhoda - komplikovaná implementace).

varianty – zápis vždy nebo podle příznaku

předběžného zápisu- kombinací metod

**organizace paměti cache:**

**nějaká varianta asociativní paměti** (V zásadě se jedná o tabulku):

|  |  |  |
| --- | --- | --- |
| **klíče (tagy)** - podle kterých se v tabulce hledá | **příslušná data** | popřípadě **služební data** (ke správě) |

Průběh hledání:vstupní hodnota klíče (argument) se najednou porovnává se všemi klíče v tabulce. Realizuje se to (většinou) jako porovnání po bitech vstupní hodnoty a každého klíče (logika z hradel XOR a NOR). Tak pracují plně asociativní paměti. Je zřejmé, že jsou použitelné pro relativně malé systémy (velké množství komparátorů, dlouhý tag).

**přímé mapování cache:** Hlavní paměť - rozdělena na třídy. Třída tvořena sadou datových položek, z nichž může být v paměti cache právě jedna.

Hledání – adresa třídy ukazuje na příslušný řádek, zde uložená hodnota (tag) se srovná s argumentem a hotovo.

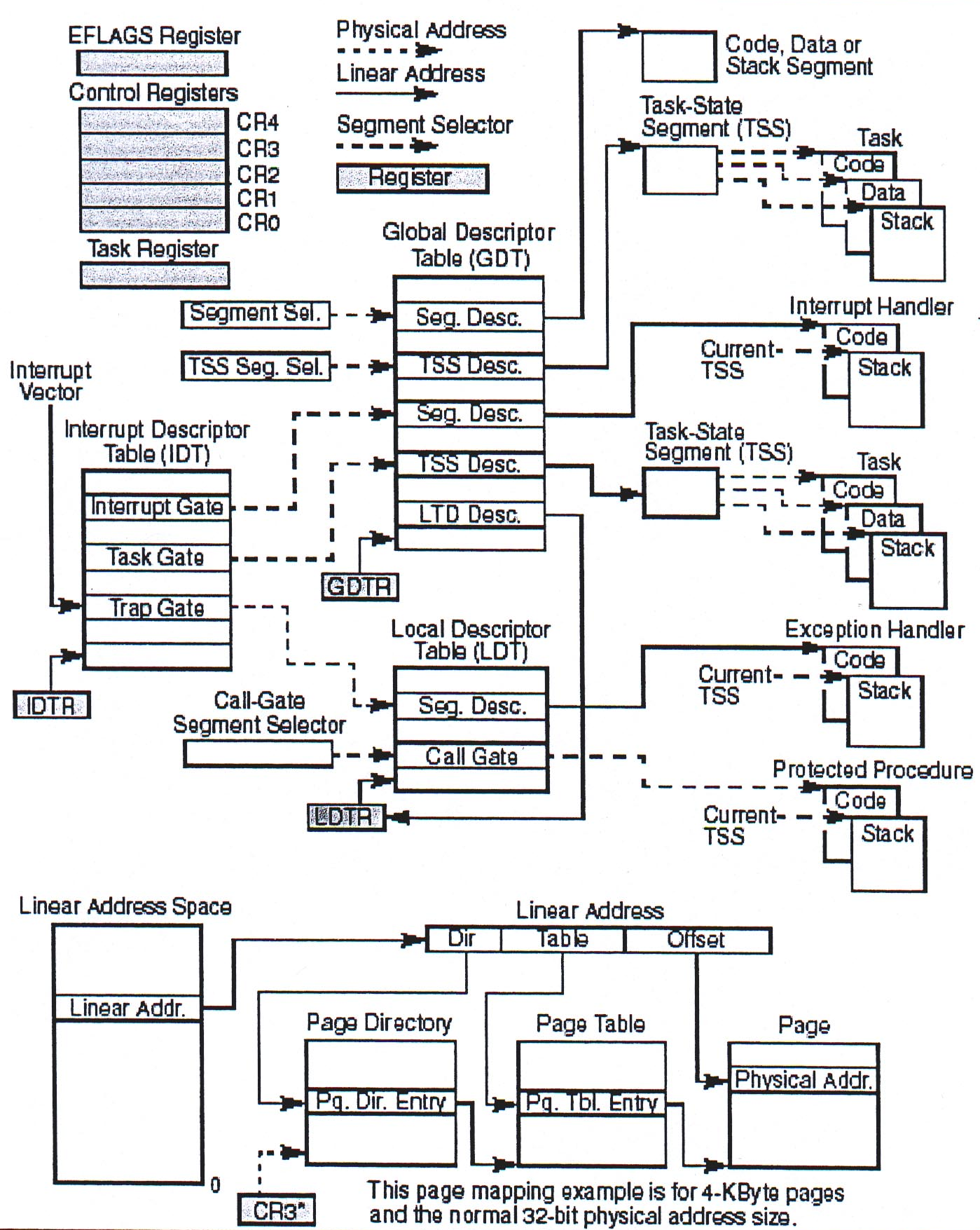
Výhoda – nejsou potřeba žádná rozhodovací kritéria, která položka bude v daném řádku paměti cache nahrazena (je tam jen jedna). Pro malé paměti cache nejefektivnější řešení. Kombinací obou přístupů

## paměť pracuje v několika režimech

pokud jsou data v cache nalezena (HIT) -> následně dodána

**nejsou-li nalezena (MISS) -> možnosti:** v cache je ještě volné místo – z hlavní paměti

přečte blok dat a příslušné slovo se dodá do CPU



Odkazy:

<http://phoenix.inf.upol.cz/esf/ucebni/OpSys.pdf>