Operačné systémy

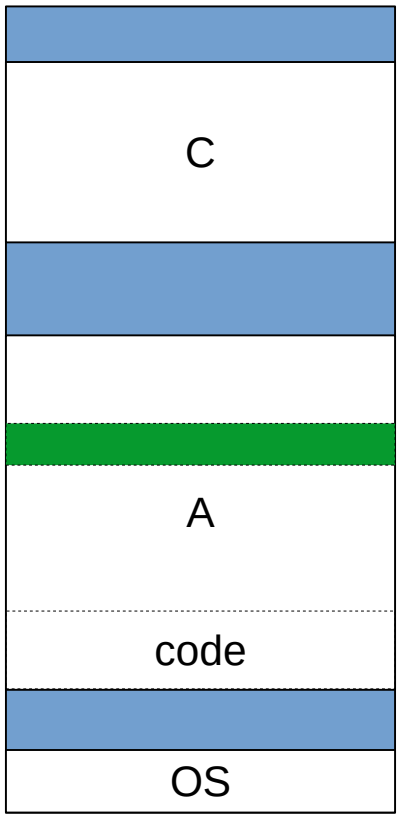
Manažment Pamäte 2. - Virtuálna pamäť

Ing. Martin Vojtko, PhD.

2024/2025

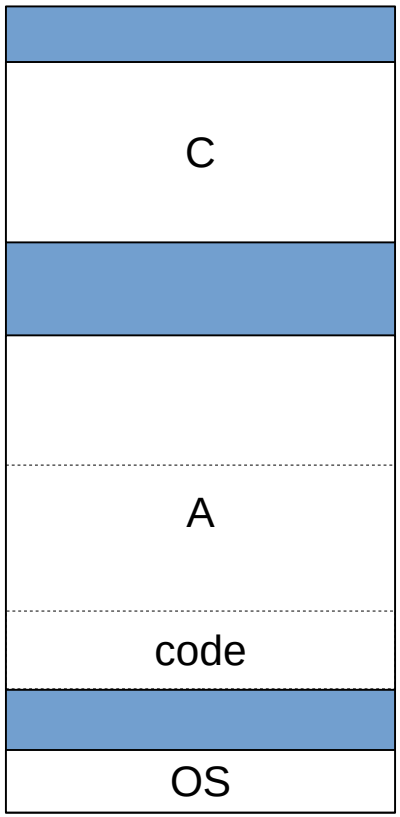
Pamäť procesu - jeden blok

- Nie je možné riešiť prístupové práva k pamäti (rwx).
- Komplikovaná alokácia novej pamäte pre zásobník alebo dáta.



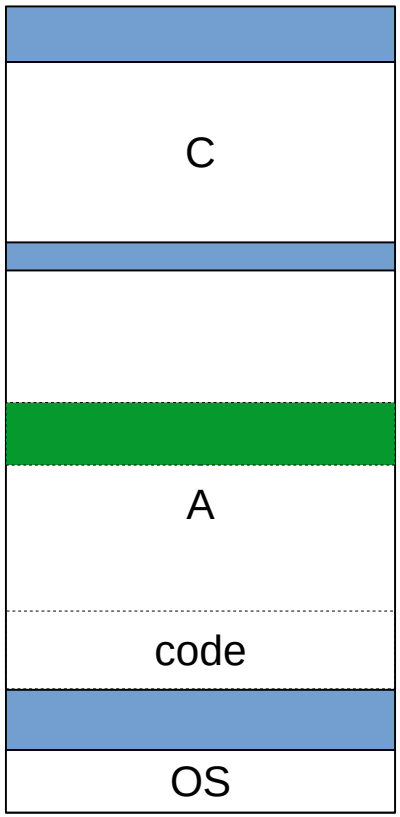
Pamäť procesu - jeden blok

- Nie je možné riešiť prístupové práva k pamäti (rwx).
- Komplikovaná alokácia novej pamäte pre zásobník alebo dáta.



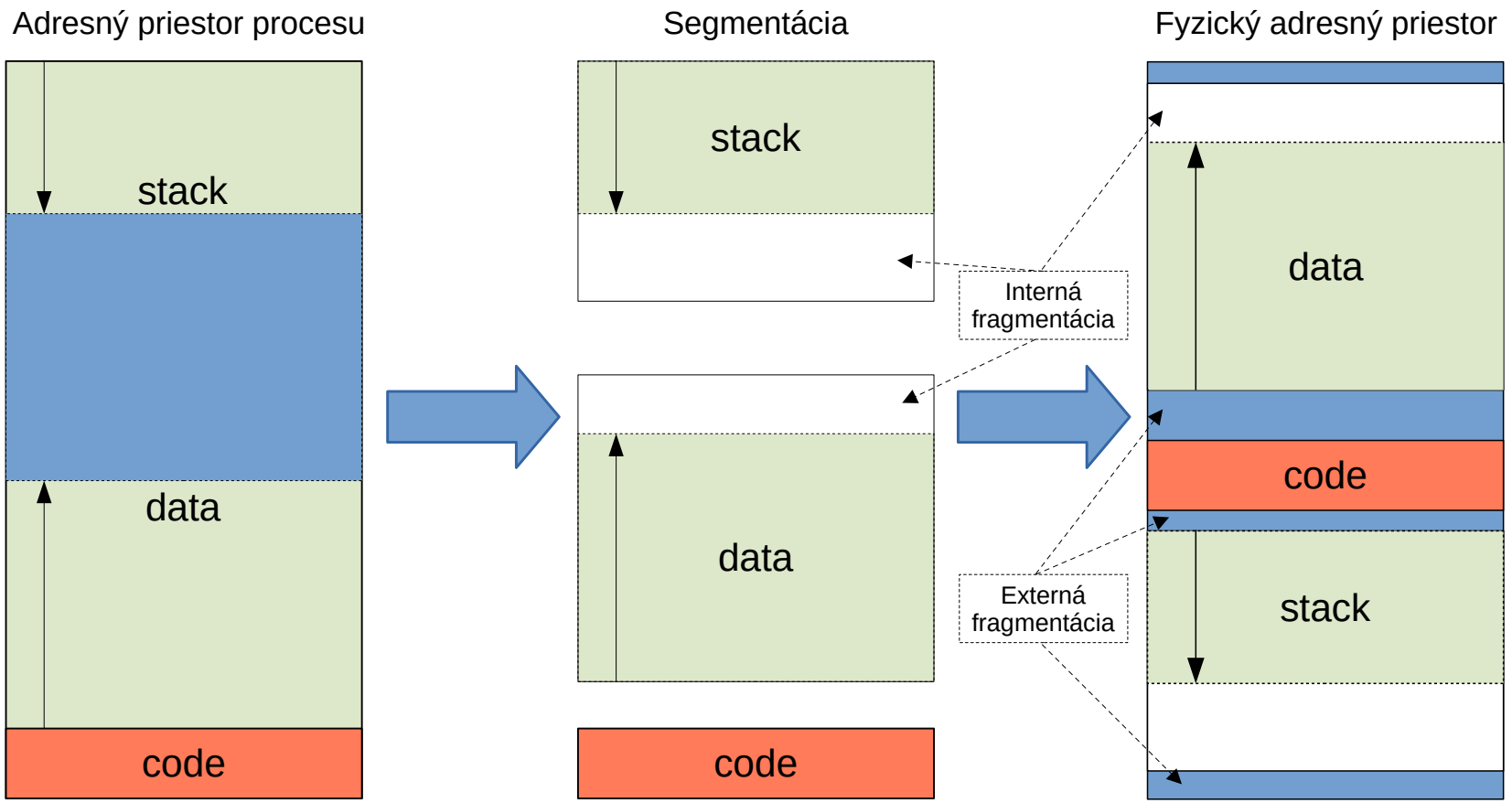
Pamäť procesu - jeden blok

- Nie je možné riešiť prístupové práva k pamäti (rwx).
- Komplikovaná alokácia novej pamäte pre zásobník alebo dáta.



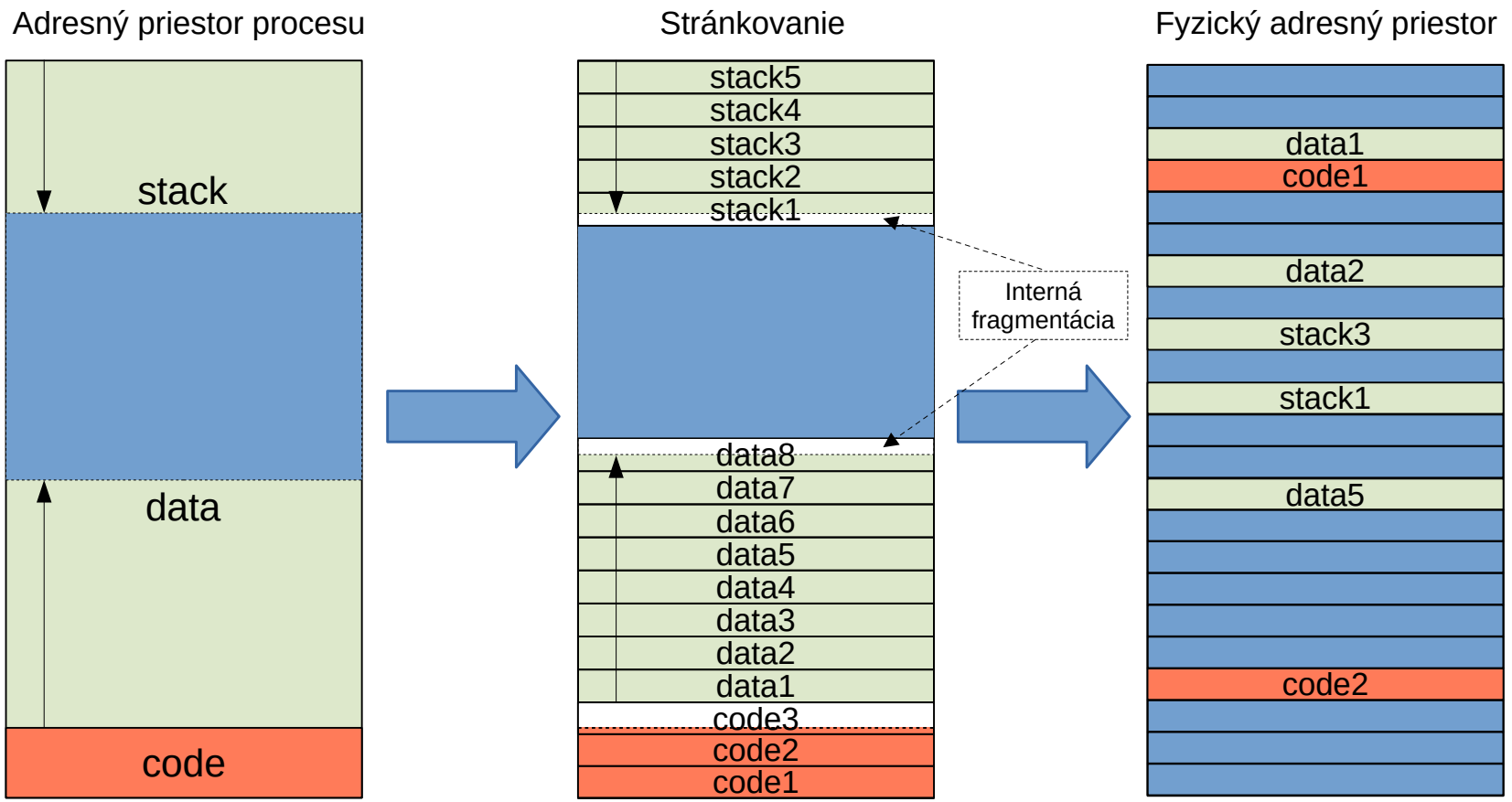
Pamäť procesu - viac blokov

- Riešením problému alokácie a práv je virtualizácia adresného priestoru procesu.



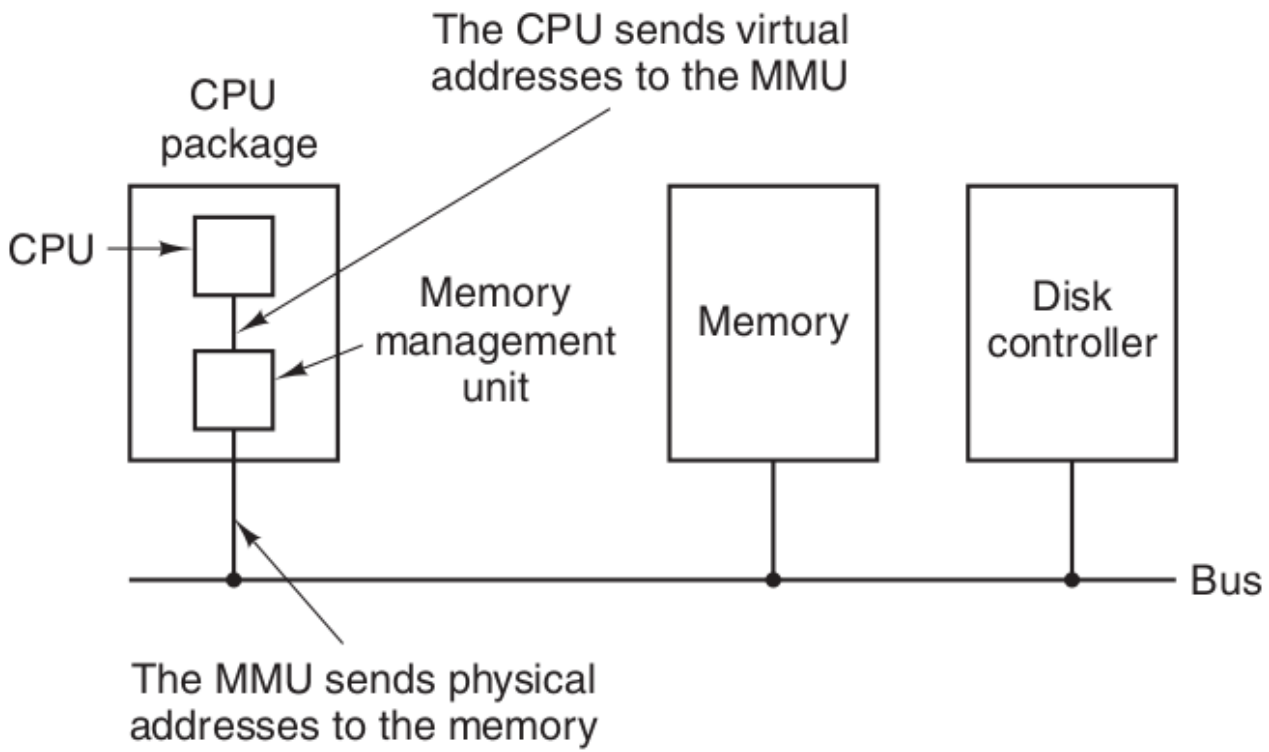
Pamäť procesu - viac blokov

- Riešením problému alokácie a práv je virtualizácia adresného priestoru procesu.



Memory Management Unit - MMU

- Pre implementáciu nutná podpora v HW - Memory Management Unit MMU



Segmentovanie

Segmentovanie

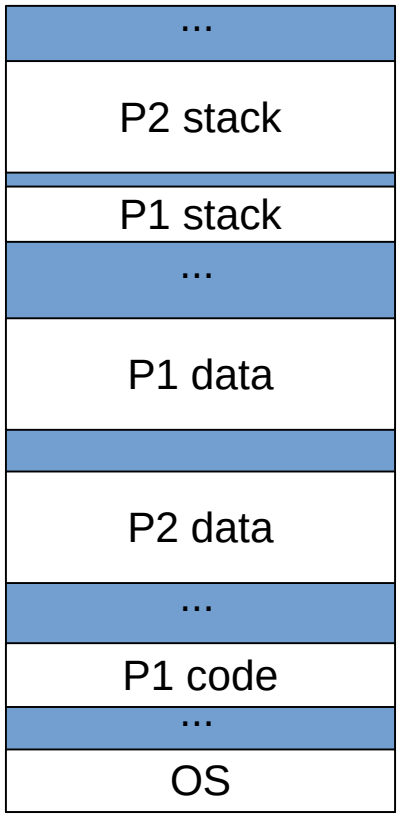
- Pamäť programu je nutné chrániť pred zápisom.
- Dáta a zásobník procesu chceme voľne a efektívne zväčšovať.

Segment

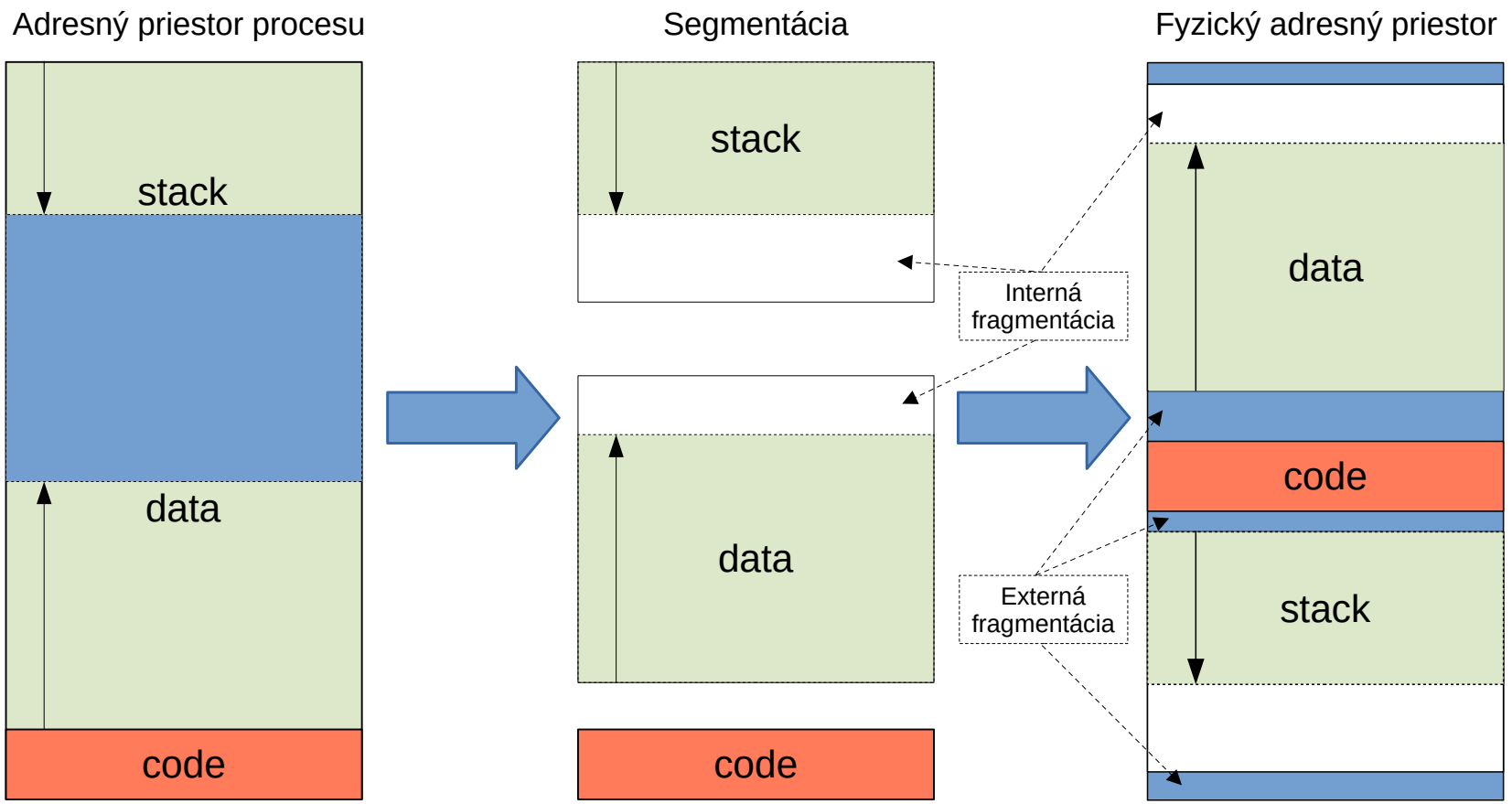
je adresný priestor vyhradený pre určitú časť pamäte procesu. Každý segment môže teoreticky zaberať celú fyzickú pamäť. Typicky jeden proces má code, data a stack segment. Segmentácia umožňuje procesu mať ľubovolné množstvo segmentov.

Segmentovanie

- Každý segment začína od adresy 0 po definovaný limit.
- Každý segment môže mať vlastné prístupové oprávnenia.
- Každý segment môže byť umiestnený na iné miesto v RAM.
- Zmena veľkosti segmentu je jednoduchšia.



Segmentovanie



Segmentovanie

- Pri swap-ovaní vieme odložiť na disk aj jeden segment procesu.
- Code segment nemusíme odkladať do swap-u.
- Je bežné, že code segment chce pracovať s dátami v dátovom segmente.
- Správne adresovanie dát v segmente zabezpečuje kompilátor.
- Segmenty sú umiestnené v pamäti RAM náhodne.
- Vyhľadanie adresy segmentu a dát v segmente zabezpečuje OS.
- OS udržuje tabuľky deskriptorov segmentov.

Segmentovanie

Descriptor

ukladá informácie o segmente procesu. Predovšetkým base adresa, limit a informácie o ochrane segmentu.

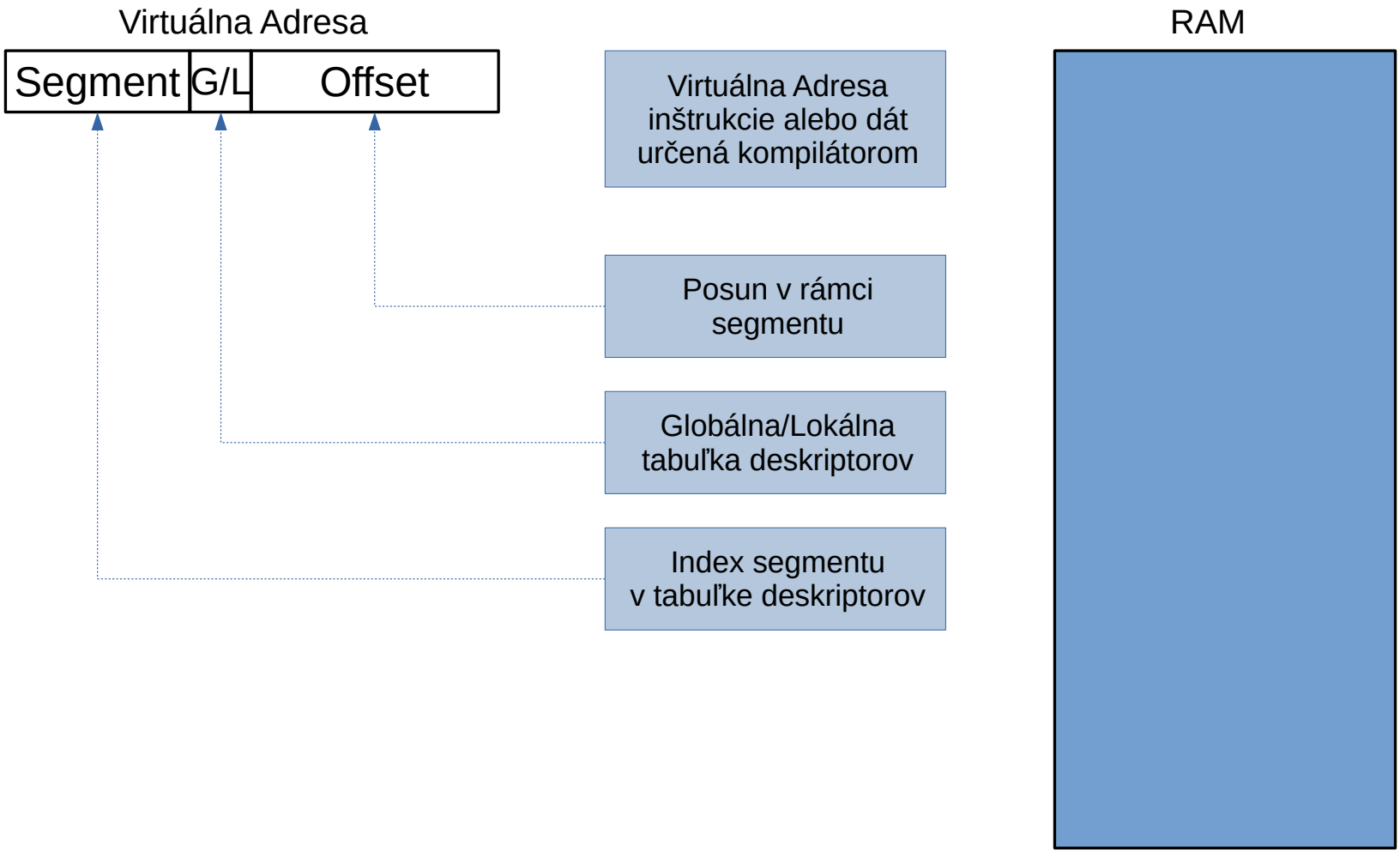
Global Descriptor Table (GDT)

je tabuľka descriptor-ov segmentov, ktoré sú prístupné pre celý výpočtový systém. Bežne sú tu udržiavané segmenty zdieľaných knižníc a segmentov OS.

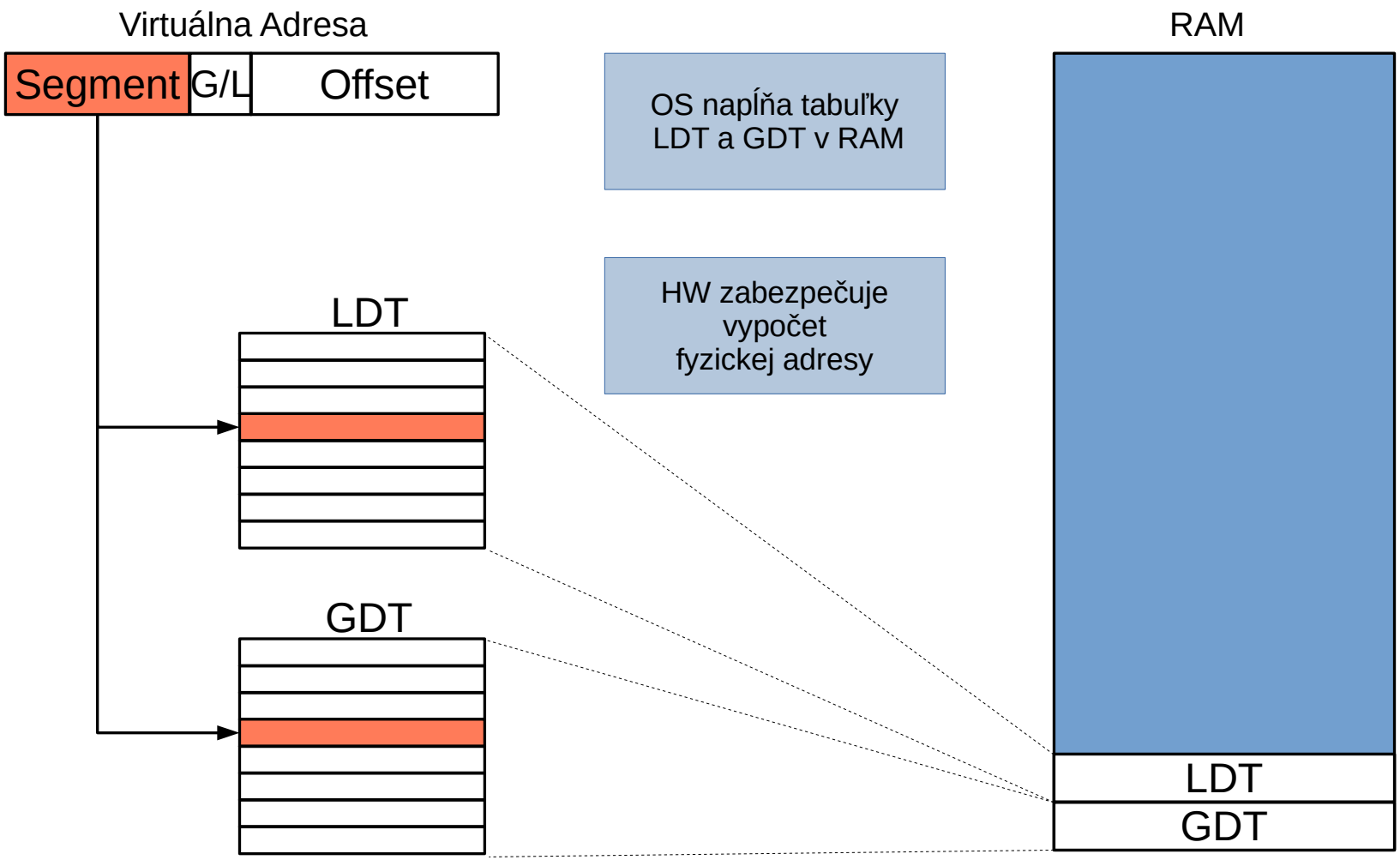
Local Descriptor Table (LDT)

je tabuľka descriptor-ov segmentov konkrétneho procesu. Pointer do LDT sa udržuje ako súčasť PCB.

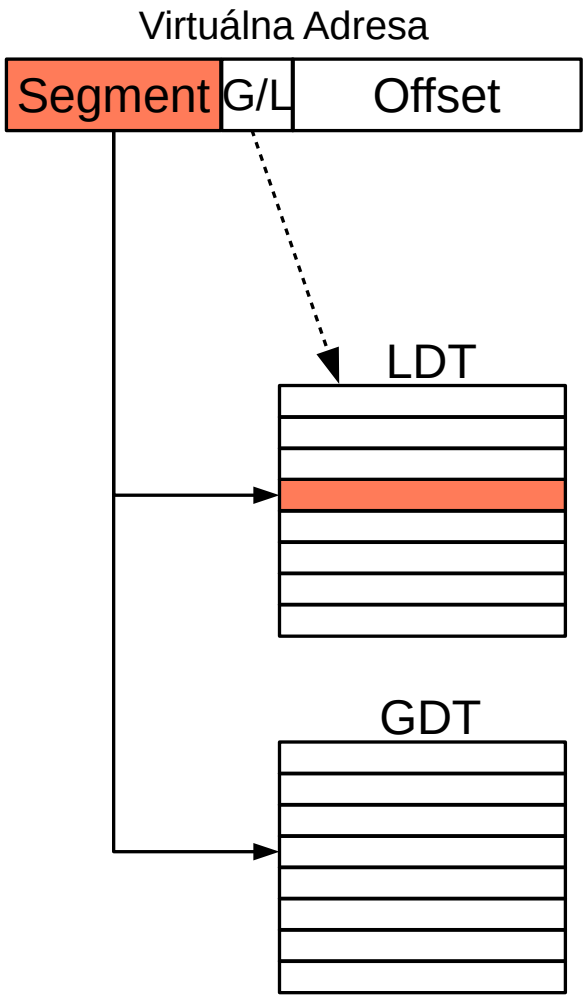
Segmentovanie



Segmentovanie



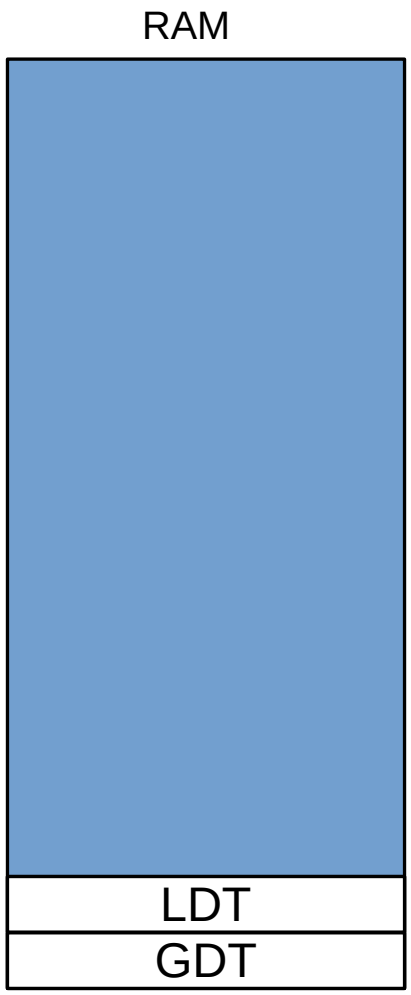
Segmentovanie



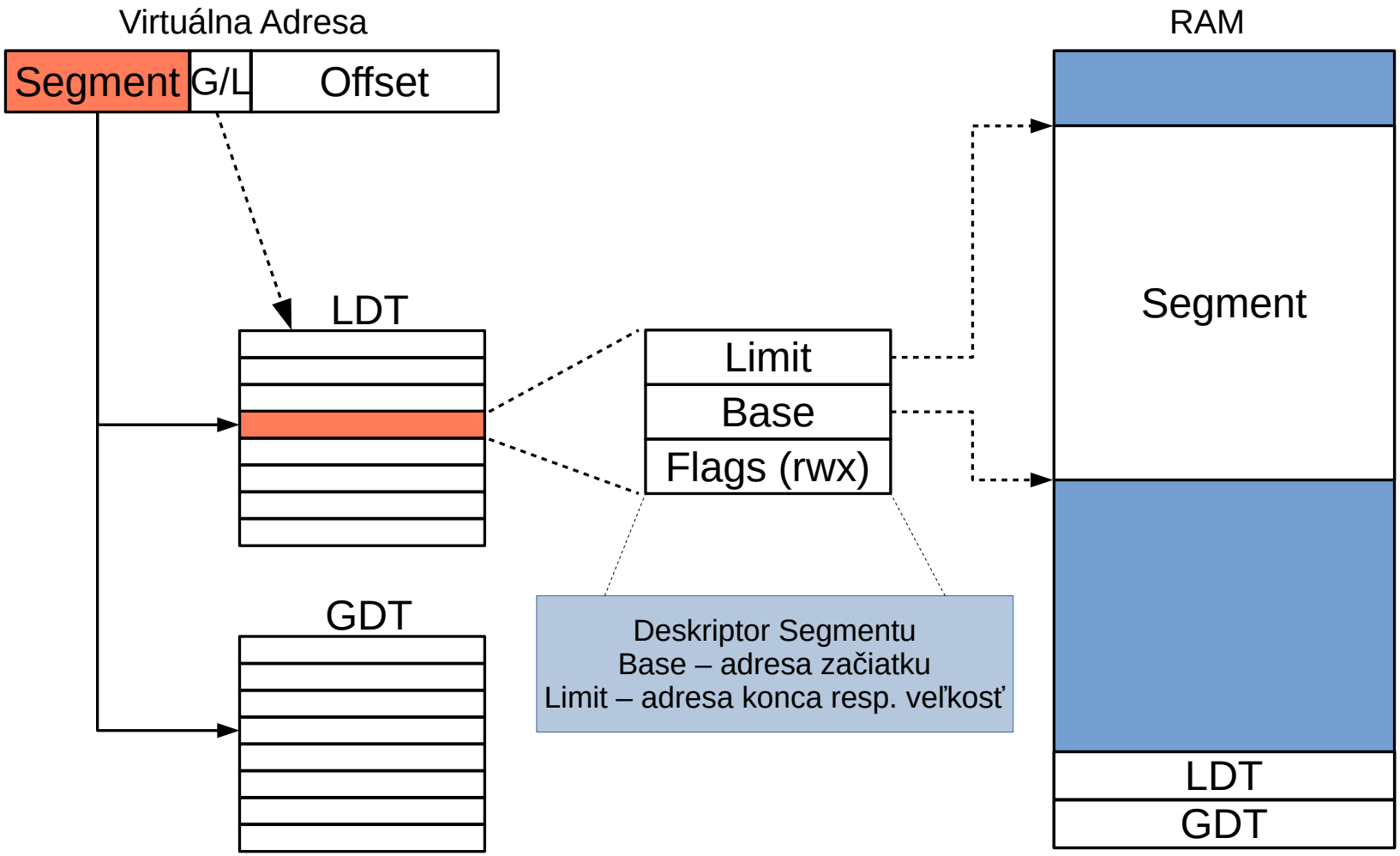
HW nájde deskriptor v RAM

Deskriptor segmentu obsahuje informáciu či segment je v RAM

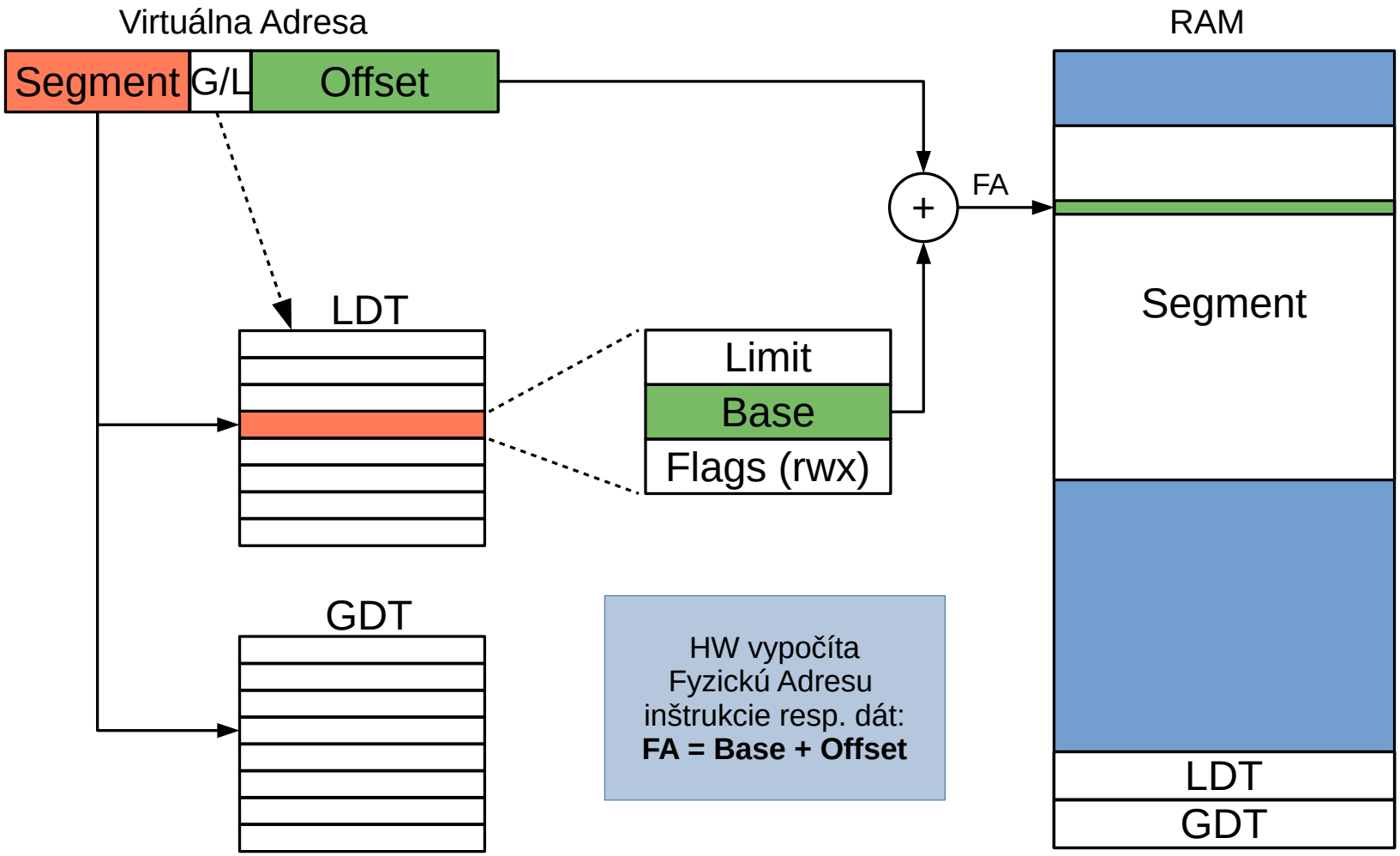
Ak nie vzniká výnimka proces je pozastavený a OS musí načítať segment do RAM



Segmentovanie



Segmentovanie



Stránkovanie

Stránkovanie

- Swap-ovanie vyžaduje odloženie celého procesu .resp segmentu na disk.
- Veľkosť procesu .resp jeho segmentov je obmedzená veľkosťou RAM.
- Priveľký proces neumožňuje multiprogramming v plnej miere.
- Potrebujeme techniku, ktorá umožní načítanie časti procesu.

Paging (Stránkovanie)

je metóda delenia pamäte procesu na rovnako veľké bloky, stránky (pages).

- Veľkosť procesu nie je obmedzená veľkosťou hlavnej pamäte.
- Stránkovanie umožňuje mať procesy obsadzujúce celý adresný priestor procesora.

Stránkovanie

Page (Stránka)

je súvislý blok pamäte procesu.

Page Frame (Stránkový rám)

je súvislý blok v hlavnej pamäti výpočtového systému. Veľkosť bloku je totožná s veľkosťou stránky.

Page Table (Tabuľka stránok)

je tabuľka OS/procesu udržujúca mapovanie stránok procesu na stránkové rámy hlavnej pamäte.

Page Fault

je situácia kedy stránka v tabuľke stránok nemá pridelený stránkový rám. HW vyvolá prerušenie, ktorým OS načíta stránku do pamäte.

Stránkovanie

- Pamäť procesu je rozdelená do stránok s rovnakou veľkosťou (napr. 4kB).
- Proces má v hlavnej pamäti iba tie stránky, ktoré potrebuje.
- Stránky umožňujú riadenie prístupu podobne ako pri segmentoch.

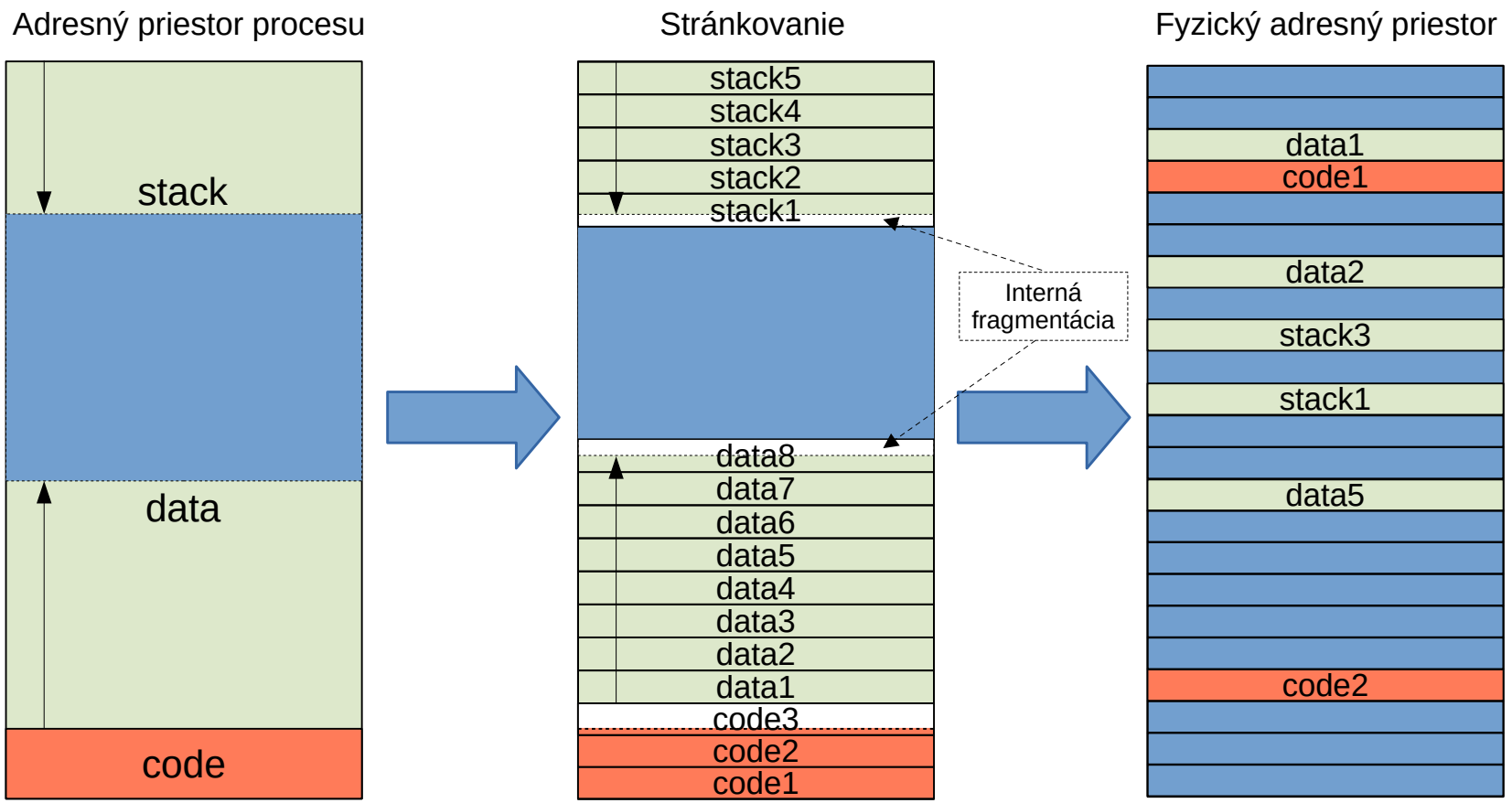
Virtual Address (Virtuálna Adresa)

je adresa elementu v adresnom priestore procesu. Pozostáva z čísla stránky a offsetu.

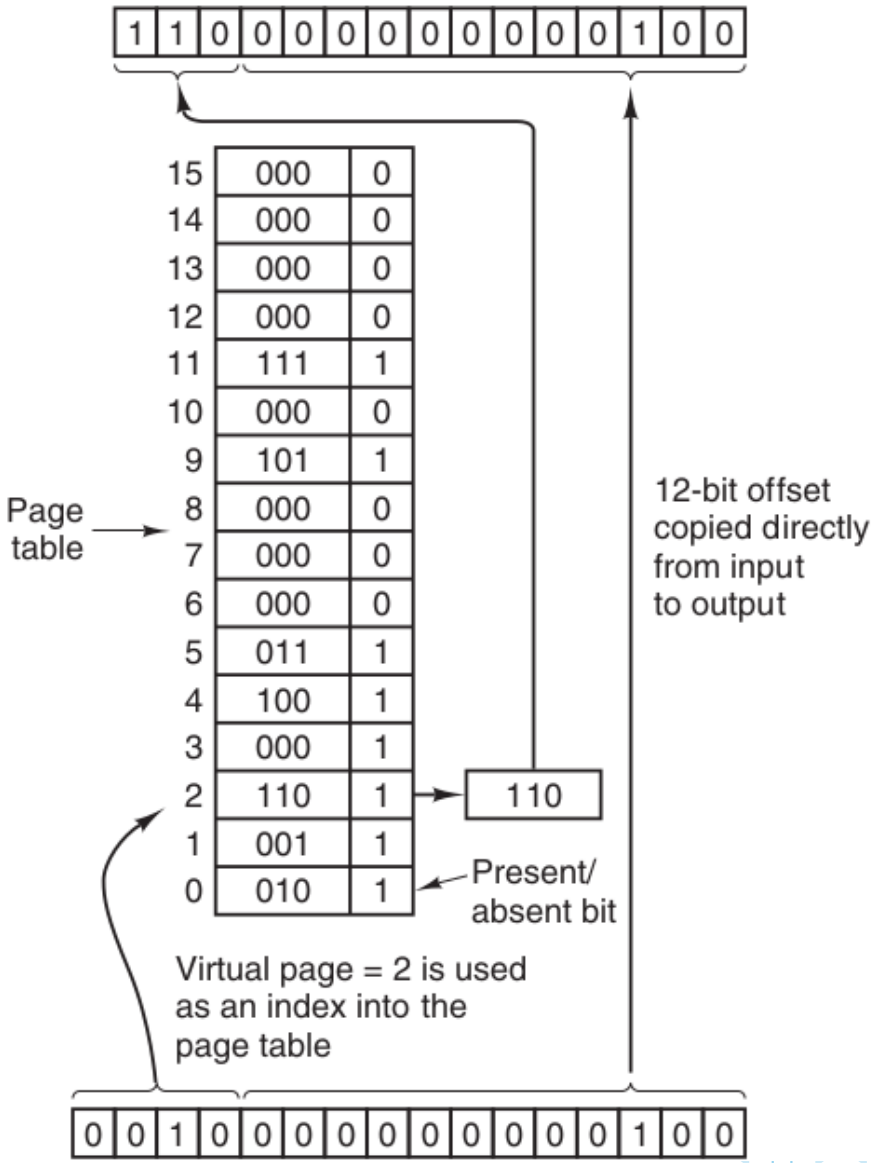
Physical Address (Fyzická Adresa)

je skutočná adresa v hlavnej pamäti výpočtového systému.

Stránkovanie

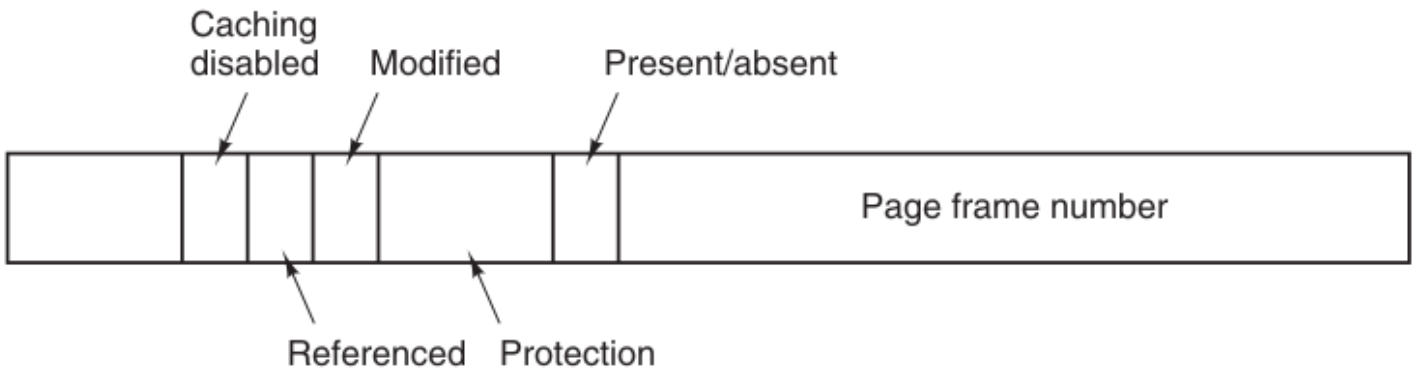


Stránkovanie - MMU



Stránkovanie - MMU

- Jeden záznam tabuľky stránok obsahuje:
  - informáciu či je stránka v hlavnej pamäti (present/absent)
  - číslo stránkového rámu (page frame number)
  - protection bity (rwx)
  - stav stránky (modified, referenced, cached)



Stránkovanie - MMU TLB

- Pri veľkých adresných priestoroch tabuľka stránok dosahuje veľké rozmery.
- Nie je možné celú tabuľku stránok udržovať v MMU.
- MMU teda obsahuje len malú asociatívnu pamäť.

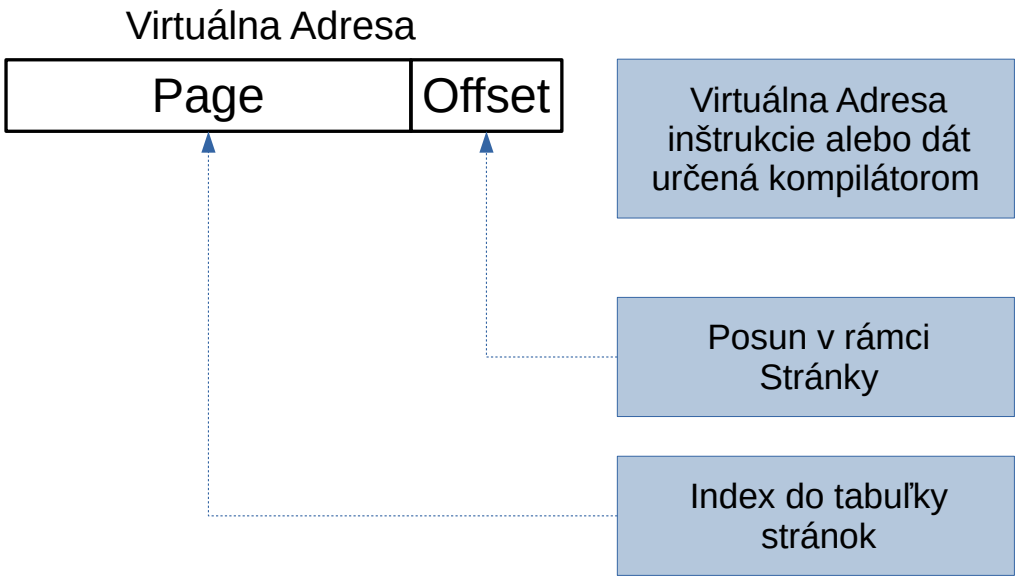
Transaction Lookaside Buffer

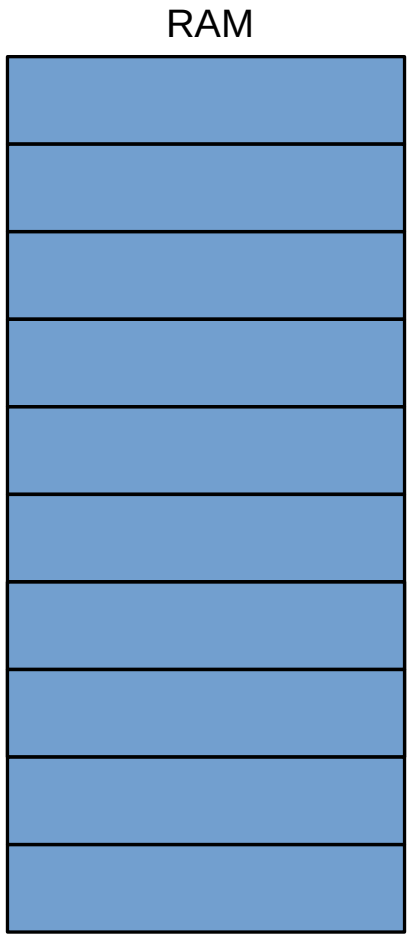
je malá asociatívna pamäť udržujúca najčastejšie používané stránky procesu. Počet položiek TLB dosahuje niekoľko desiatok.

Stránkovanie - MMU TLB

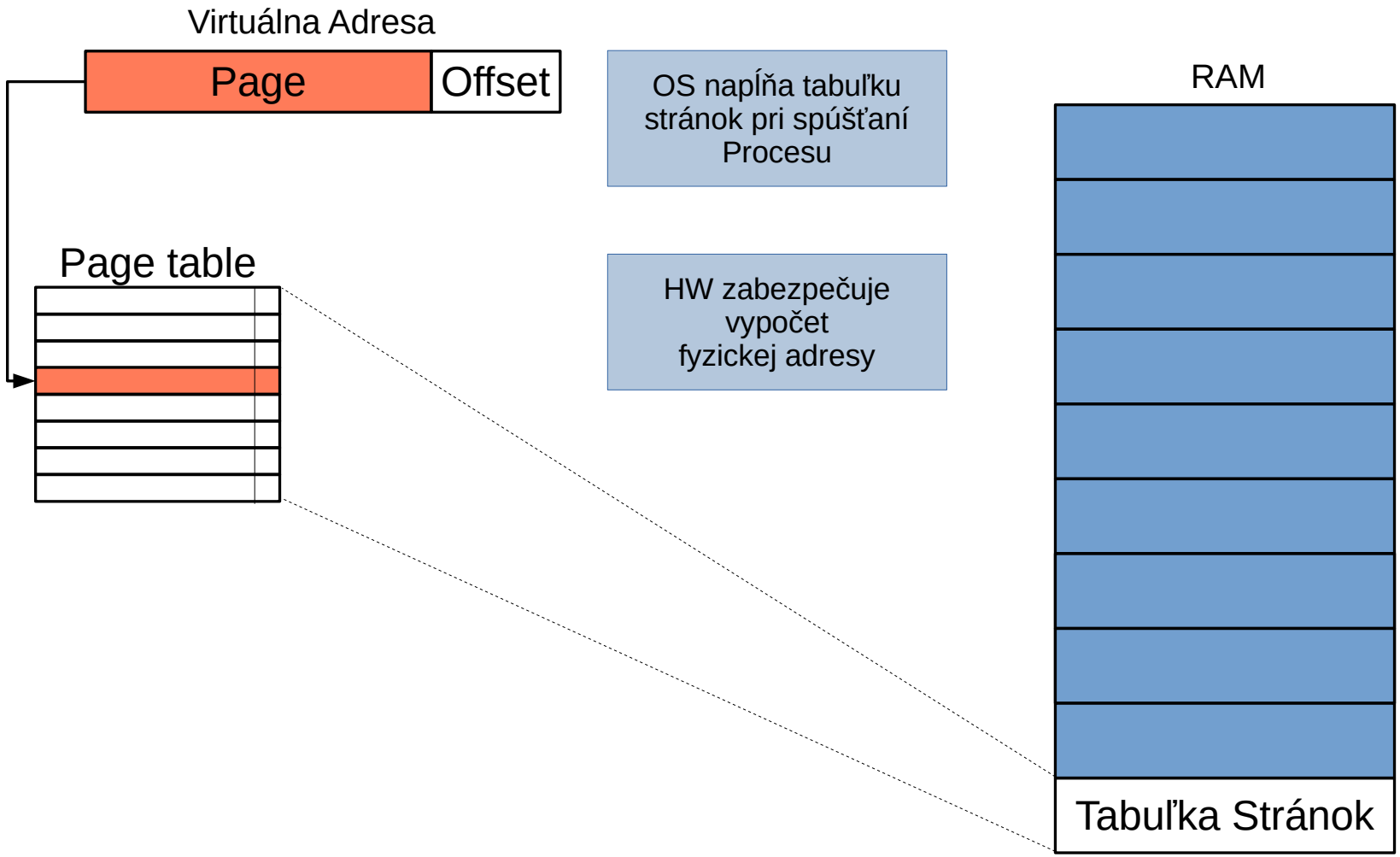
- Pri výpočte fyzickej adresy sa MMU pozrie či stránka nemá záznam v TLB.
- Ak áno máme vyhrané. Ak nie hovoríme o tzv. soft/tlb page fault.
- Ak záznam nebol v TLB MMU sa pozrie do tabuľky stránok, ktorá sa už nachádza v hlavnej pamäti.
- Ak stránka má pridelený stránkový rám máme vyhrané. Ak nie máme page fault.
- Môže sa stať, že aj pri hľadaní tabuľky stránok procesu dôjde k page fault.

Stránkovanie - jedna úroveň

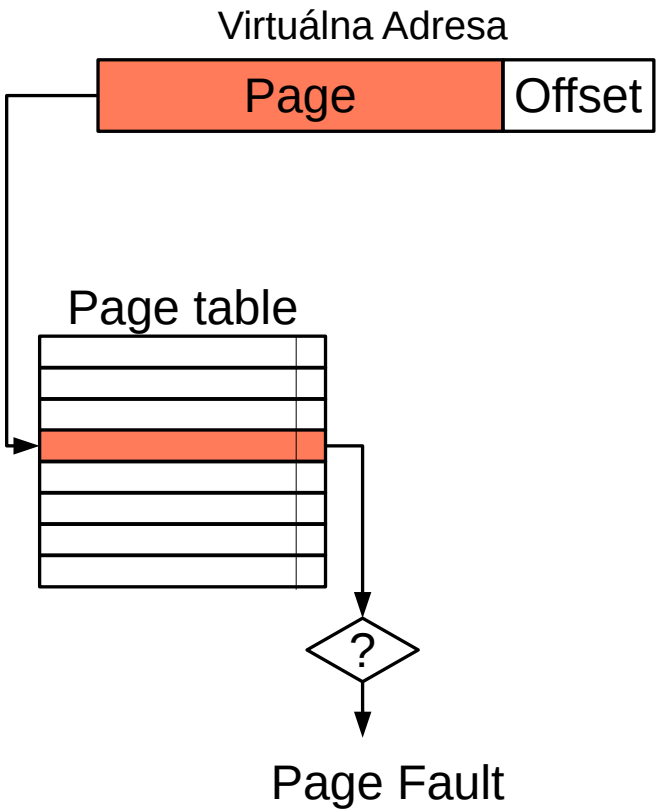




Stránkovanie - jedna úroveň



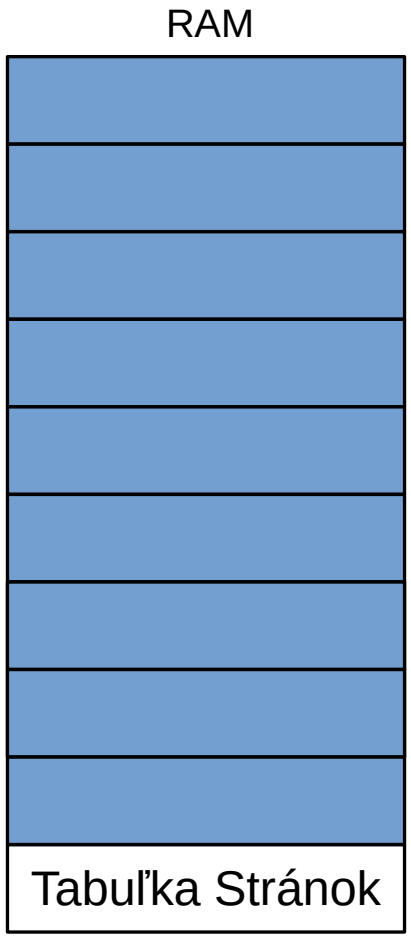
Stránkovanie - jedna úroveň



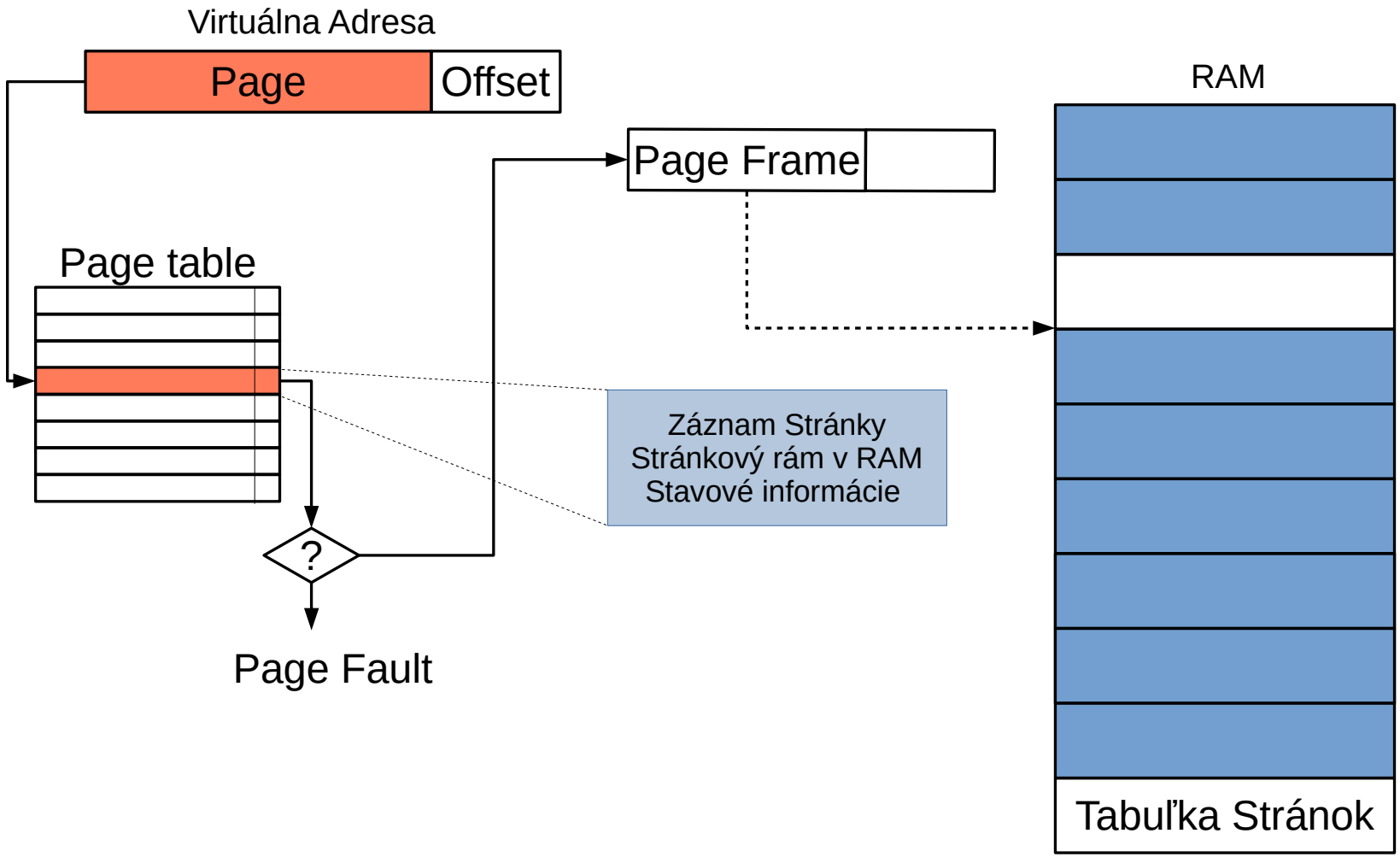
HW nájde záznam o stránke v RAM

Záznam stránky obsahuje informáciu či stránka je v RAM

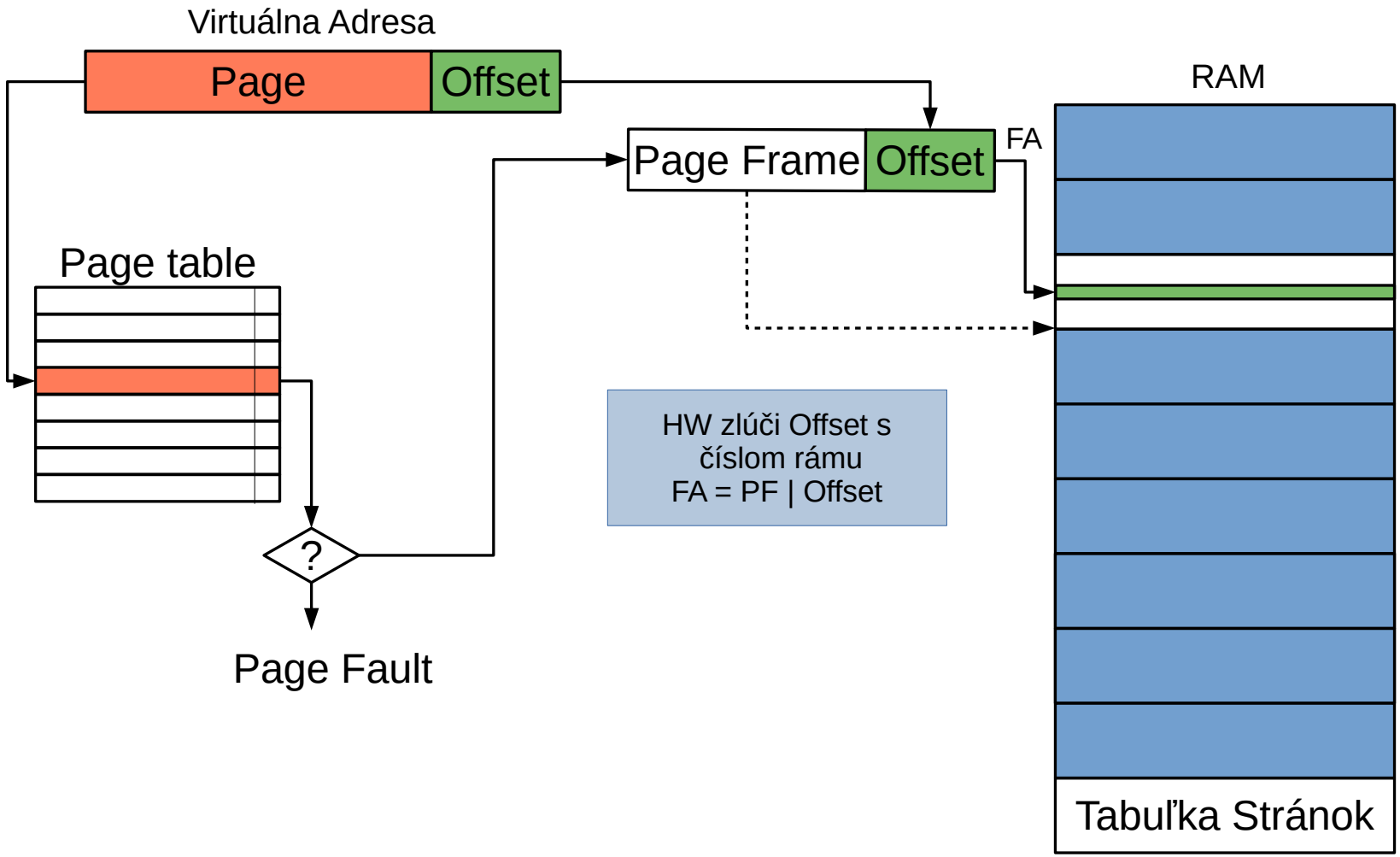
Ak nie vzniká výnimka proces je pozastavený a OS musí načítať stránku do RAM



Stránkovanie - jedna úroveň



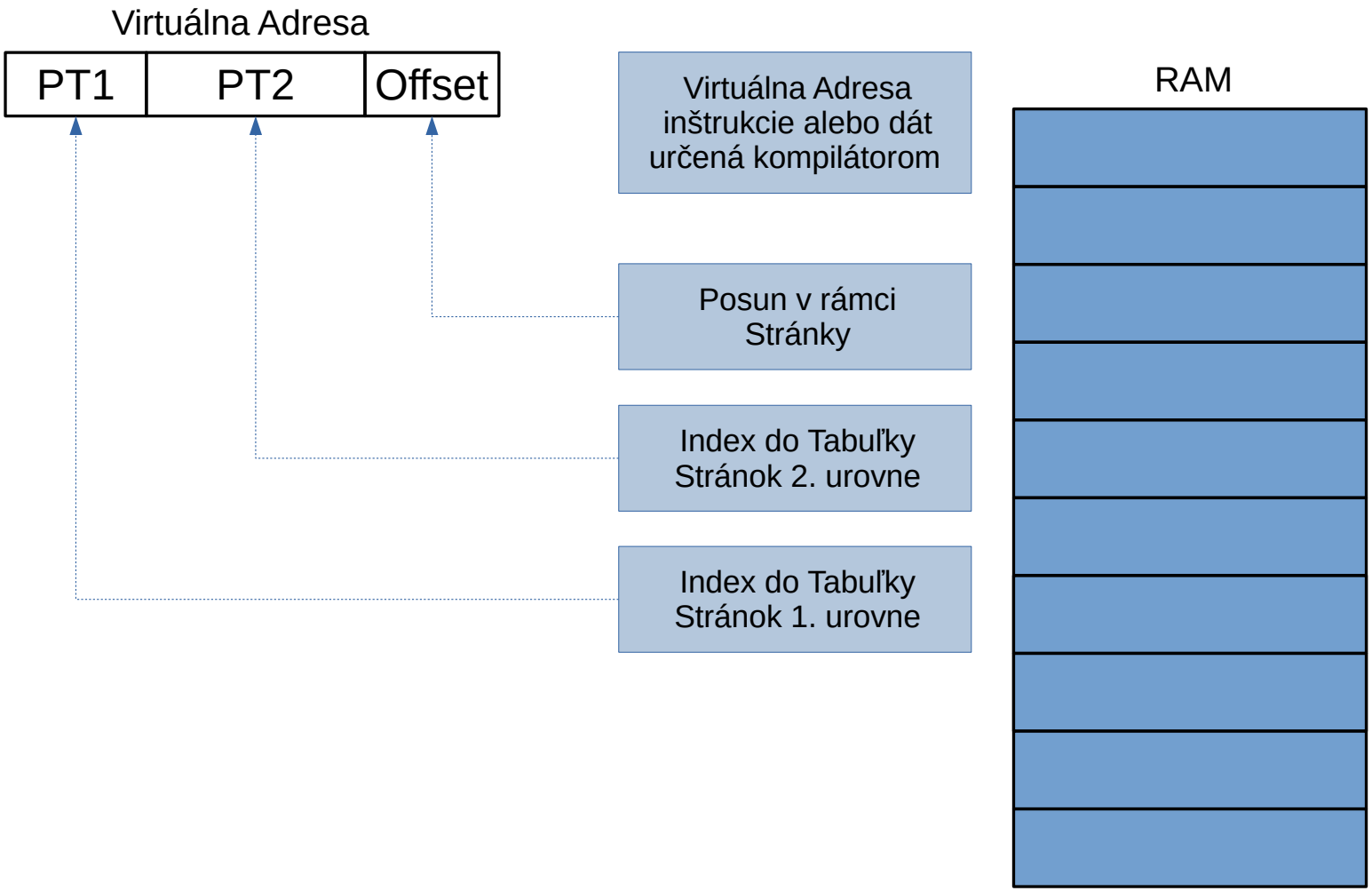
Stránkovanie - jedna úroveň



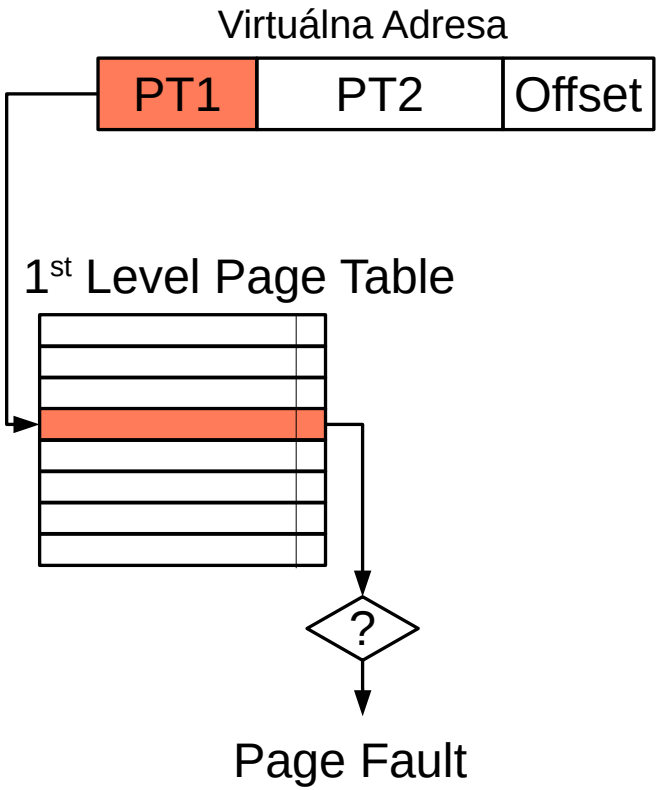
Stránkovanie - jedna úroveň

- Bežná veľkosť stránky je $4kB = 2^{12}B$. Čo je 1K 32b slov.
- Pri 32b adrese máme $2^{32}/2^{12} = 2^{20}$ stránok.
- Tabuľka stránok má 1M záznamov = 4MB.
- Ak mám program, ktorého samotná veľkosť je 64 kB po vytvorení procesu bude mať daný proces v pamäti 4MB + 64kB.

Stránkovanie - dve úrovne



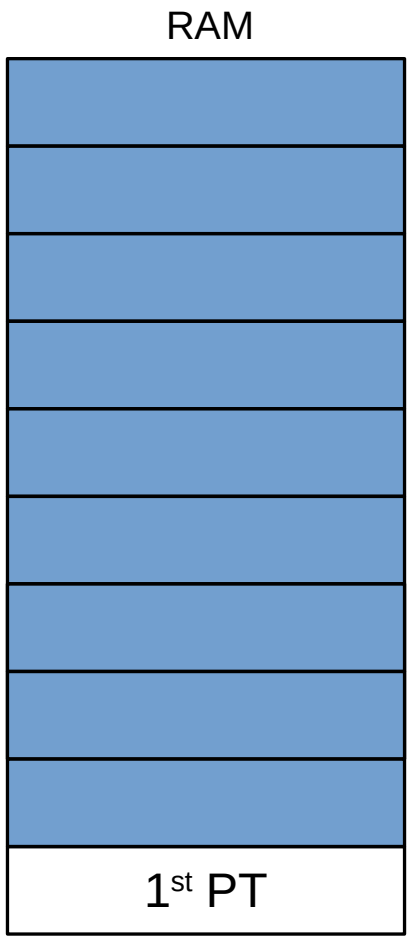
Stránkovanie - dve úrovne



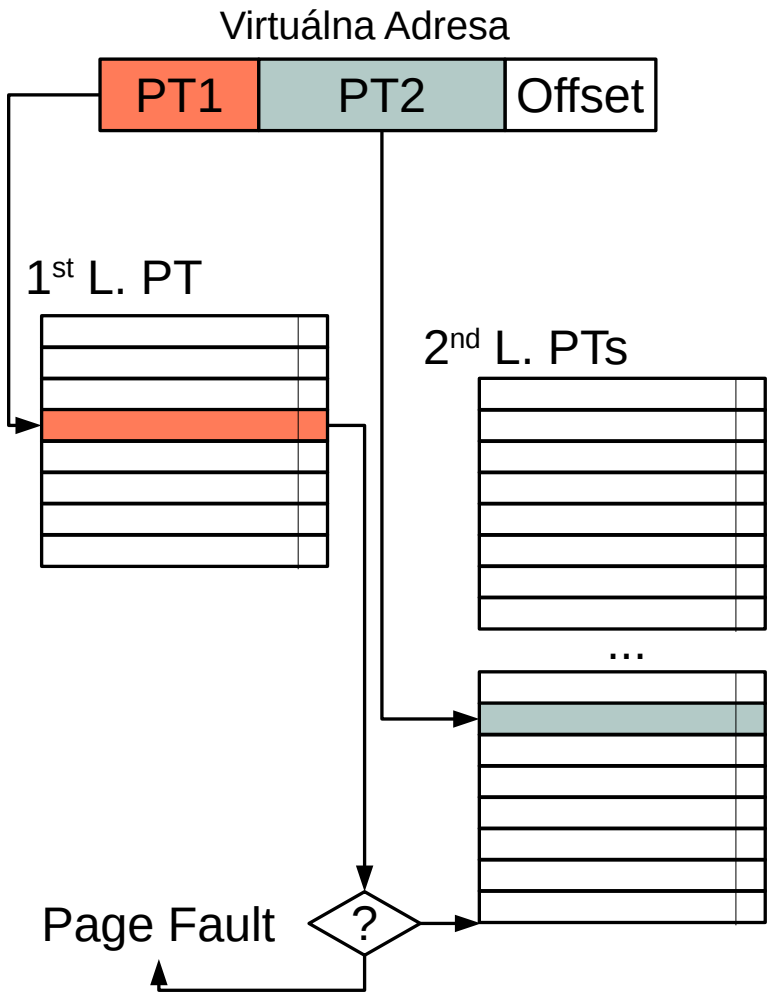
HW nájde záznam o tabuľke v RAM

Záznam tabuľky obsahuje informáciu či tabuľka je v RAM

Ak nie vzniká výnimka proces je pozastavený a OS musí načítať tabuľku do RAM



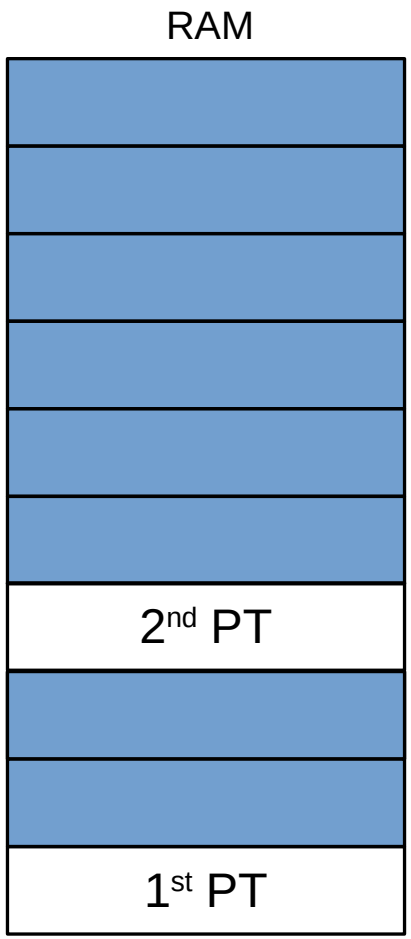
Stránkovanie - dve úrovne



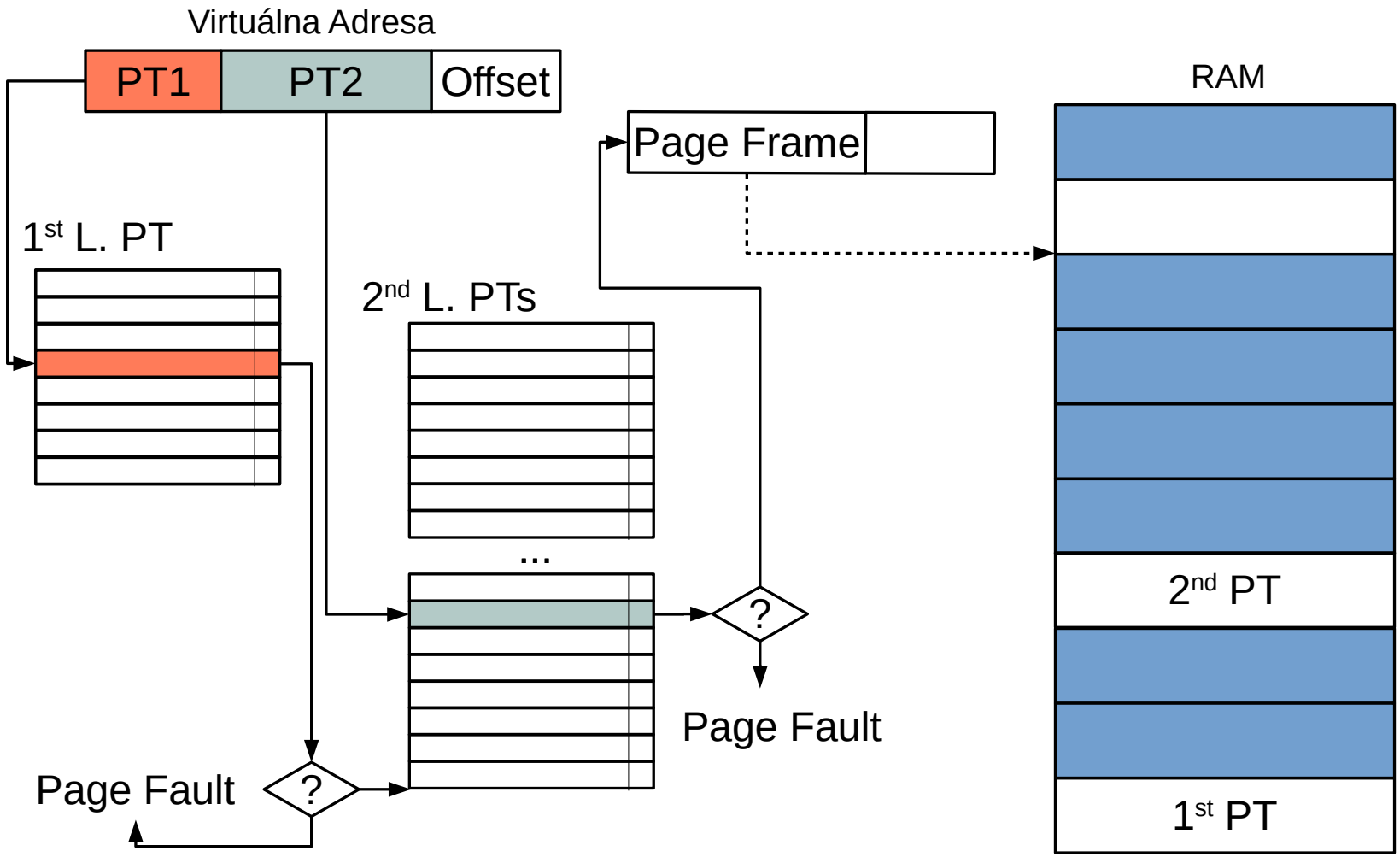
HW nájde záznam o stránke v RAM

Záznam stránky obsahuje informáciu či stránka je v RAM

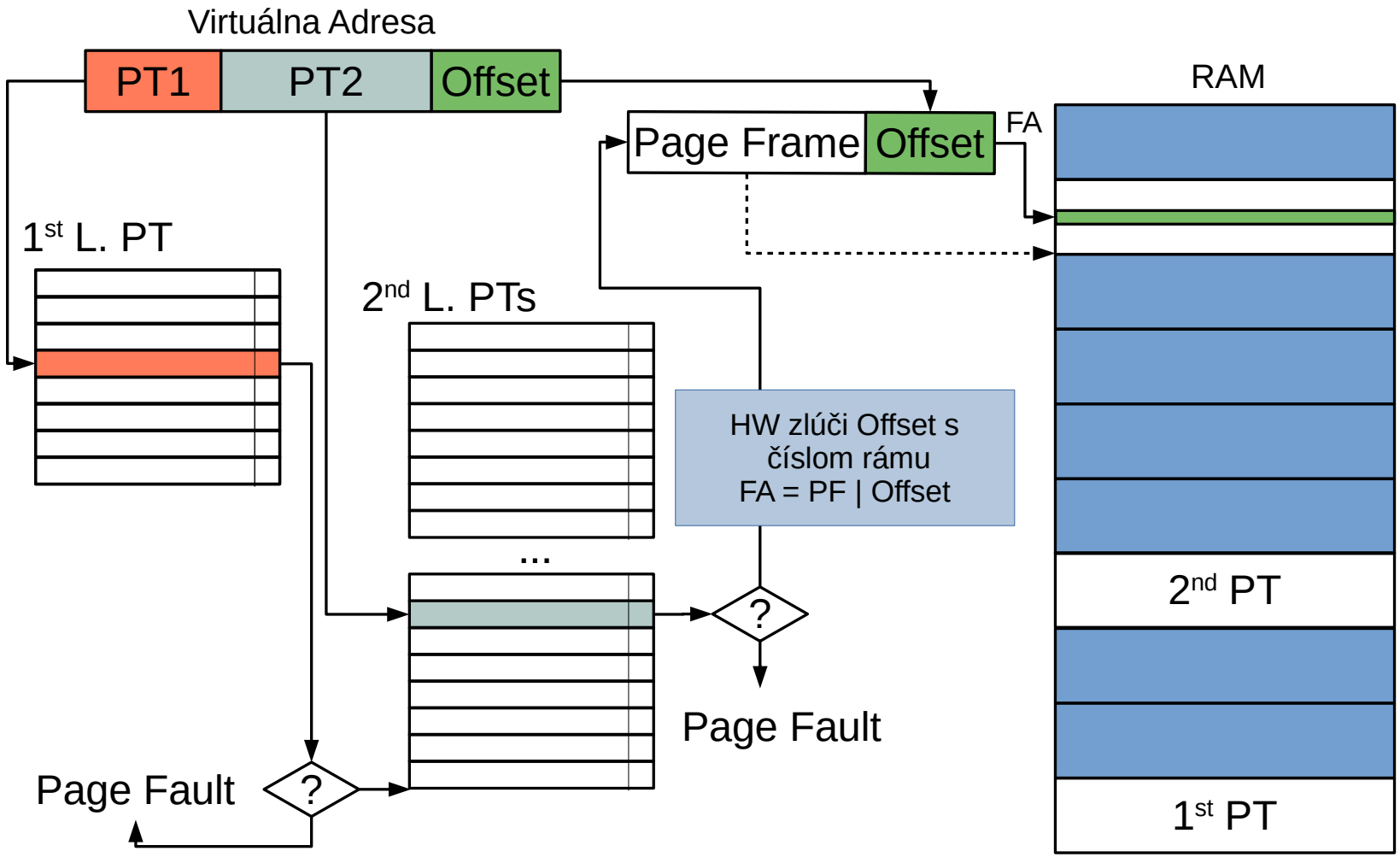
Ak nie vzniká výnimka proces je pozastavený a OS musí načítať stránku do RAM



Stránkovanie - dve úrovne



Stránkovanie - dve úrovne



Stránkovanie - dve úrovne

- Bežná veľkosť stránky je $4kB = 2^{12}B$. Čo je 1k 32b slov.
- Adresa rozdelená do dvoch úrovní po 10b
- Prvá úroveň tabuľka tabuliek stránok má $2^{10} = 1024$ položiek = 4kB.
- Položka prvej úrovne ukazuje na tabuľku stránok rovnako 1k položiek = 4kB.
- Jedna tabuľka stránok adresuje $2^{22}B = 4MB$ pamäte.
- Najmenší proces bude potrebovať jedu tabuľku tabuliek a jednu tabuľku stránok = 8 kB + veľkosť programu.

Stránkovanie - viac úrovní

- Dvoj úrovňové stránkovanie je nepostačujúce pri výpočtových systémoch s 48b alebo 64b adresnými zbernicami.
- Pri 48b zbernici a 4kB stránkach máme 2^{36} stránok = 64GB.
- Pri 48b zbernici a 4kB stránkach a dvoch úrovniach 2^{26} tabuliek stránok = 64MB.
- Bežne sa používajú 3 alebo 4 úrovne stránkových tabuliek.
- Bežne sa používajú väčšie stránky.

Stránkovanie a segmentovanie

Stránkovanie a segmentovanie

Stránkovanie a Segmentovanie

Stránkovanie s TLB a Segmentovaním

Dvoj úrovňové Stránkovanie s TLB a Segmentovaním

Viac úrovňové Stránkovanie s TLB a Segmentovaním

Stránkovanie a segmentovanie

32b systémy

mohli adresovať 4GB pamäte. Mnohé programy však potrebovali väčší adresný priestor, ako bolo možné mať. Kombinácia stránkovania so segmentovaním bolo nutným riešením. Ak rozdelíme program na 3 segmenty, tak môžeme mať procesy dosahujúce 12GB.

64b systémy

bežne používajú 48b alebo až 64b adresy. Adresný priestor je obrovský, a preto segmenty už nie sú potrebné v rámci procesu. Použitie segmentovania si nájde uplatnenie pri zdielaní kódu knižnic.

Stránkovanie a segmentovanie

Consideration	Paging	Segmentation	Combination
Need the programmer be aware that this technique is being used?	No	Yes * compiler hide this	Yes *
How many linear address spaces are there?	1	Many	Many
Can the total address space exceed the size of physical memory?	Yes	Yes * segment maximum size is FA size	Yes
Can memory used by process exceed the size of address space?	No	Yes	Yes
Can procedures and data be distinguished and separately protected?	No * There are no code pages	Yes	Yes
Is sharing of procedures between users facilitated?	No	Yes	Yes
Why was this technique invented?	To get a large linear address space without having to buy more physical memory	To allow programs and data to be broken up into logically independent address spaces and to aid sharing and protection	

Page Fault

Page Fault

Soft/TLB Page Fault

je situácia kedy MMU nenájde mapovanie stránky na stránkový rám v TLB. V takom prípade MMU konzultuje tabuľku stránok uloženú v RAM.

- MMU má uloženú informáciu kde sa tabuľka stránok vykonávaného procesu nachádza.
- Soft Page Fault je celý spracovaný hardvérom.
- V prípade, že stránka v tabuľke stránok nemá pridelený stránkový rám HW vyvolá prerušenie.

Page Fault

Page Fault

je HW prerušenie CPU vyvolané MMU, ktoré nenašlo stránku v TLB a ani v tabuľke stránok. Počas Page Fault sa zavolá prerušovací pod-program OS, ktorý zabezpečí vyhľadanie chýbajúcej stránky na disku. V prípade, že v hlavnej pamäti nie je dosť miesta OS zabezpečí aj výber stránky (obete), ktorá bude vyhostená.

- Proces je vykonávaný CPU inštrukciu za inštrukciou.
- Každá nová inštrukcia je čítaná z pamäte.
- Inštrukcie majú svoje parametre, ktoré tiež často vyžadujú prístup do pamäte.
- Akékoľvek čítanie pamäte je vyhodnotené pomocou MMU.

Page Fault

Vykonanie inštrukcie

Keďže bežná inštrukcia má niekoľko parametrov, môže sa stať, že dáta každého parametru sa nachádzajú v inej stránke. Vedie to k tomu, že súčasne pristupujeme k stránke programu a n stránkam dát. Navyše (aby to nebolo jednoduché) ešte k stránkam obsahujúcim tabuľky stránok ak sme mali smolu v TLB. Jedna inštrukcia môže vyvolať niekoľko Page Fault-ov. Vykonanie inštrukcie sa nám môže predĺžiť z ns na ms (6-8 rádov).

Vyber obete

Ak zahodíme stránky, ktoré sa používajú často, tak výpočtový systém nedosahuje takú priepustnosť ako by mohol. Preto výber správneho algoritmu je veľmi dôležitý.

Page Fault

Efektívny/Priemerný čas vykonania inštrukcie

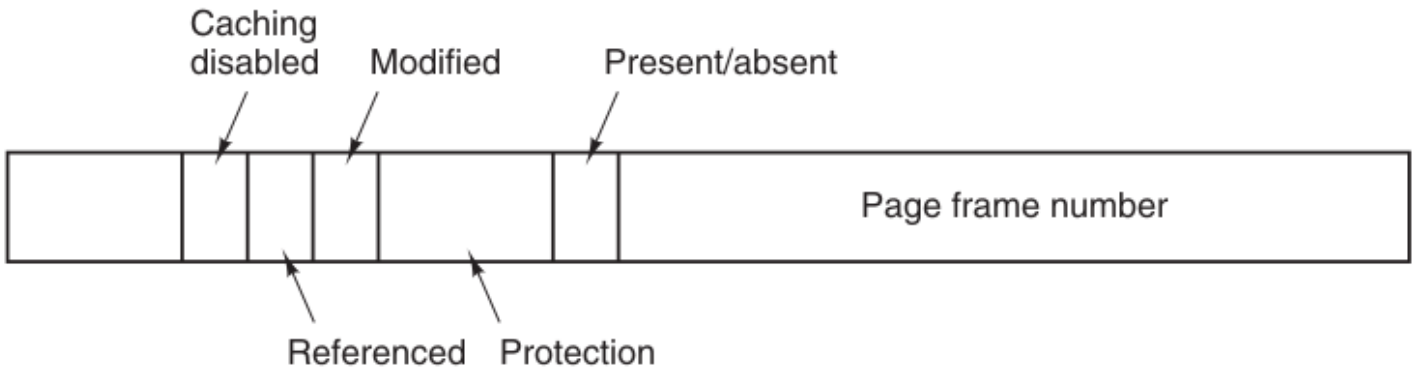
Majme n priemerný čas prečítania stránky z pamäte a k priemerný počet vykonaných inštrukcií bez Page Fault-u.
Priemerný čas vykonania k inštrukcií je: $T_k = k + n$.
Priemerný čas vykonania 1 inštrukcie je: $T_1 = \frac{T_k}{k} = \frac{k+n}{k}$

$$T_1 = 1 + \frac{n}{k}$$

- Snahou manažmentu pamäte je dosiahnuť $T_1 = 1$
  - $n \to 0$: rýchlejší disk a zbernice. (Väčšie stránky pri HDD)
  - $k \to \infty$: cpu cache, väčšia hlavná pamäť, **Optimálny algoritmus stránkovania**.

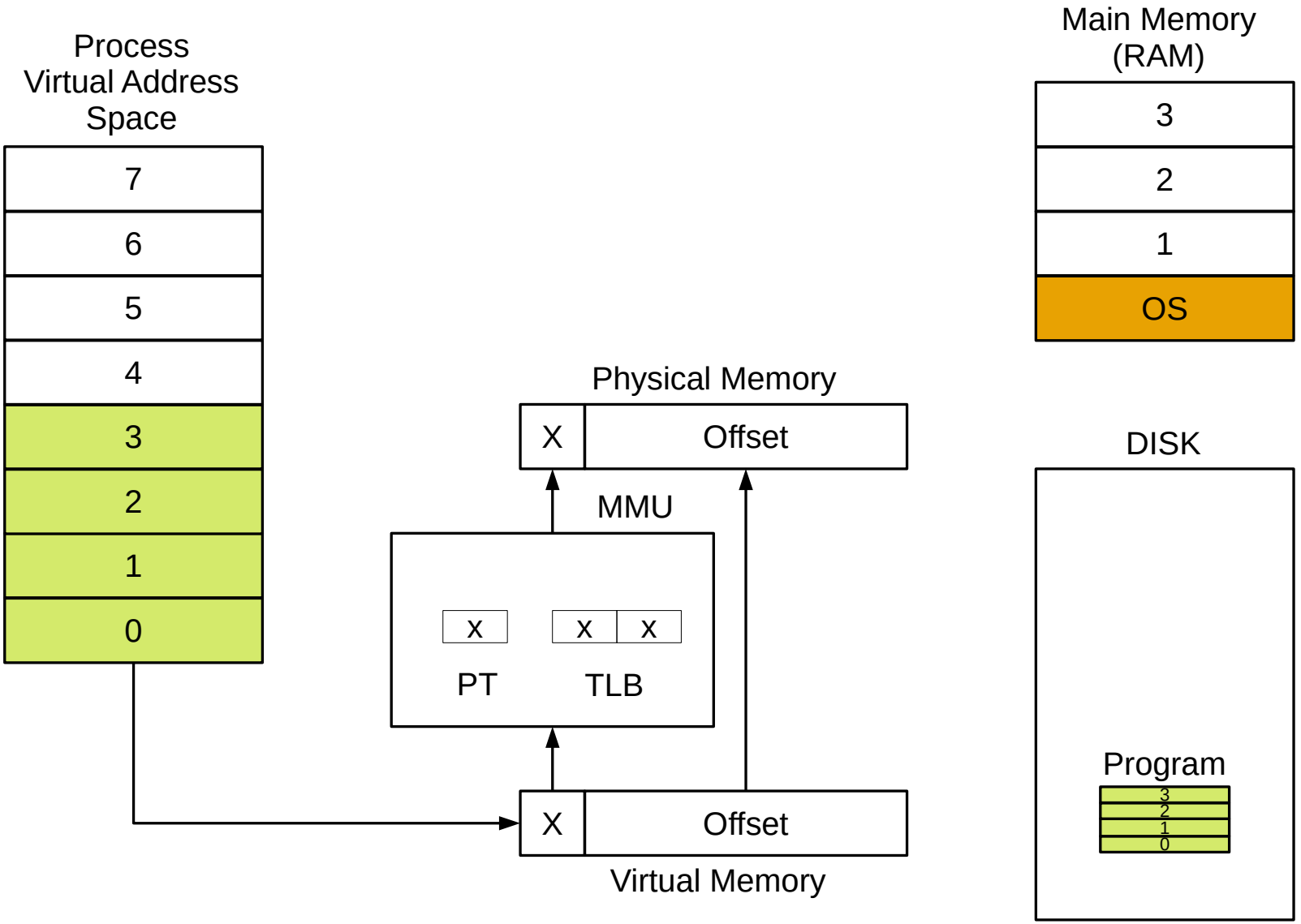
Stavové informácie o stránke

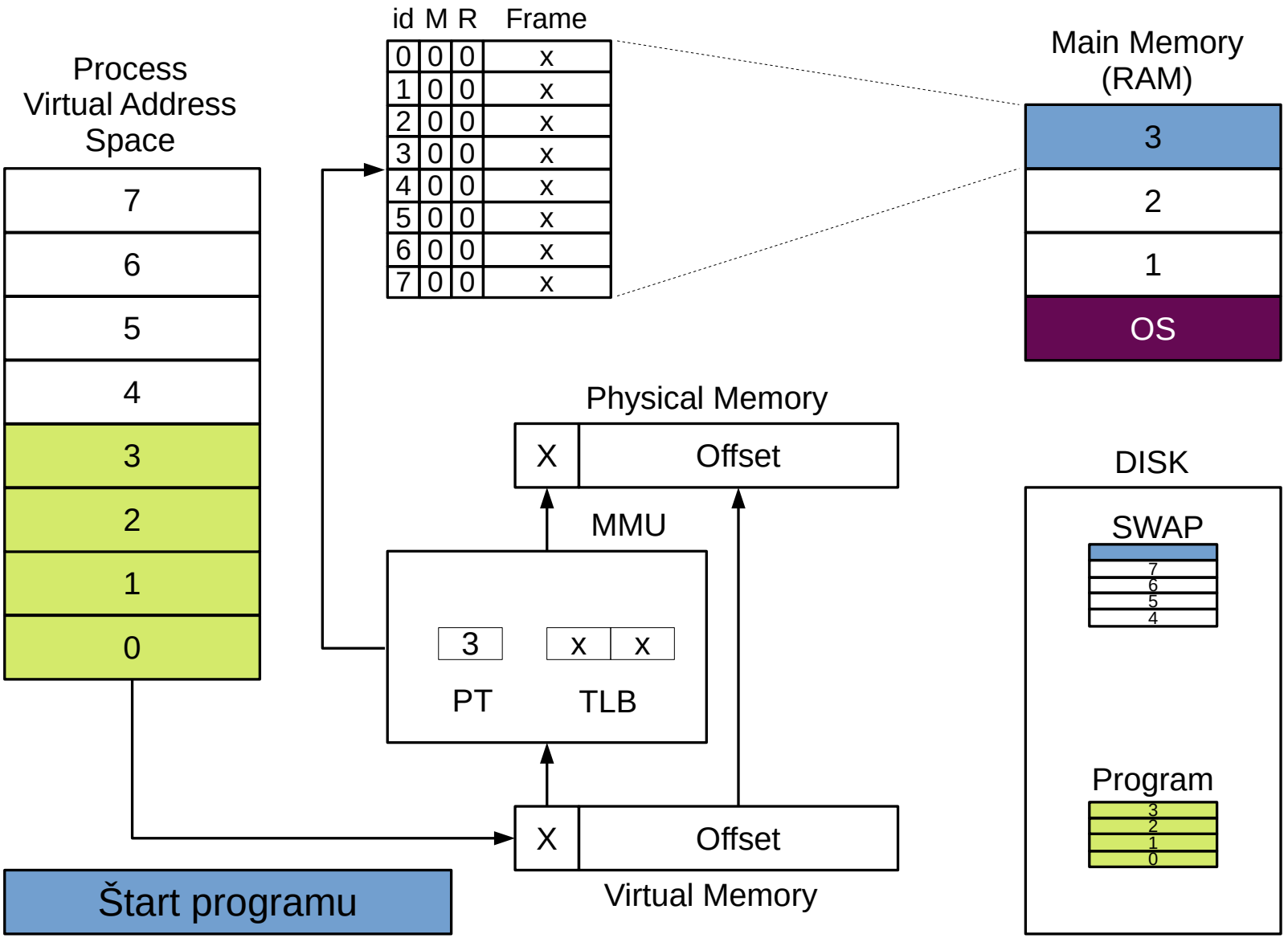
- OS a HW udržuje stav každej stránky procesu.
- Stav je odložený v položkách tabuľky stránok.
- Niektoré stavové informácie o stránke:
  - Protection bits - read, write, execute oprávnenia.
  - Present - P-bit - stránka má/nemá pridelený stránkový rám.
  - Referenced - R-bit - stránka bola referencovaná.
  - Modified - M-bit - stránka bola modifikovaná.
- Stav sa nastaví pri každom prístupe do stránky pomocou HW.
- OS vykonáva reset stavu stránky ak je potrebné.

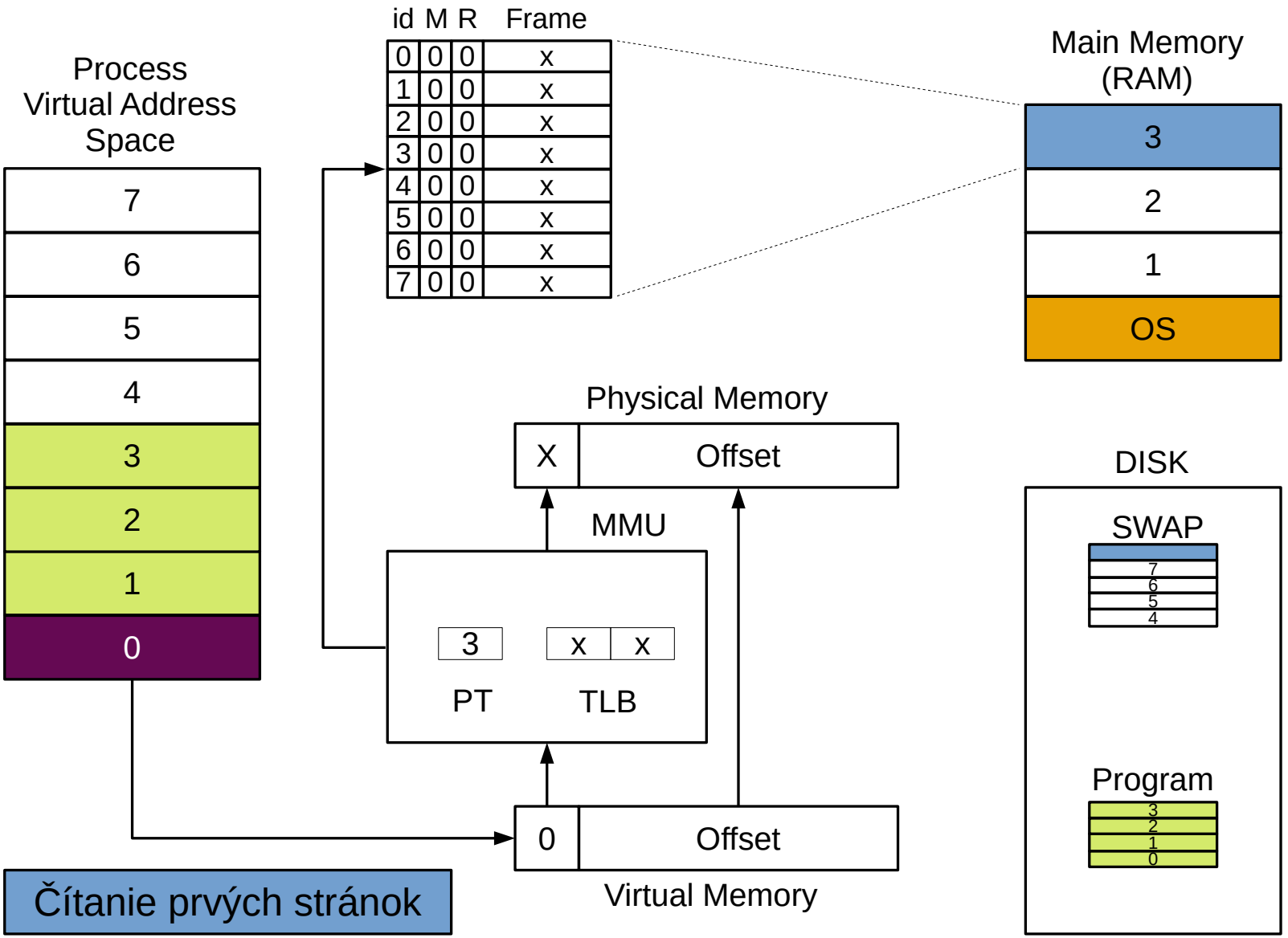


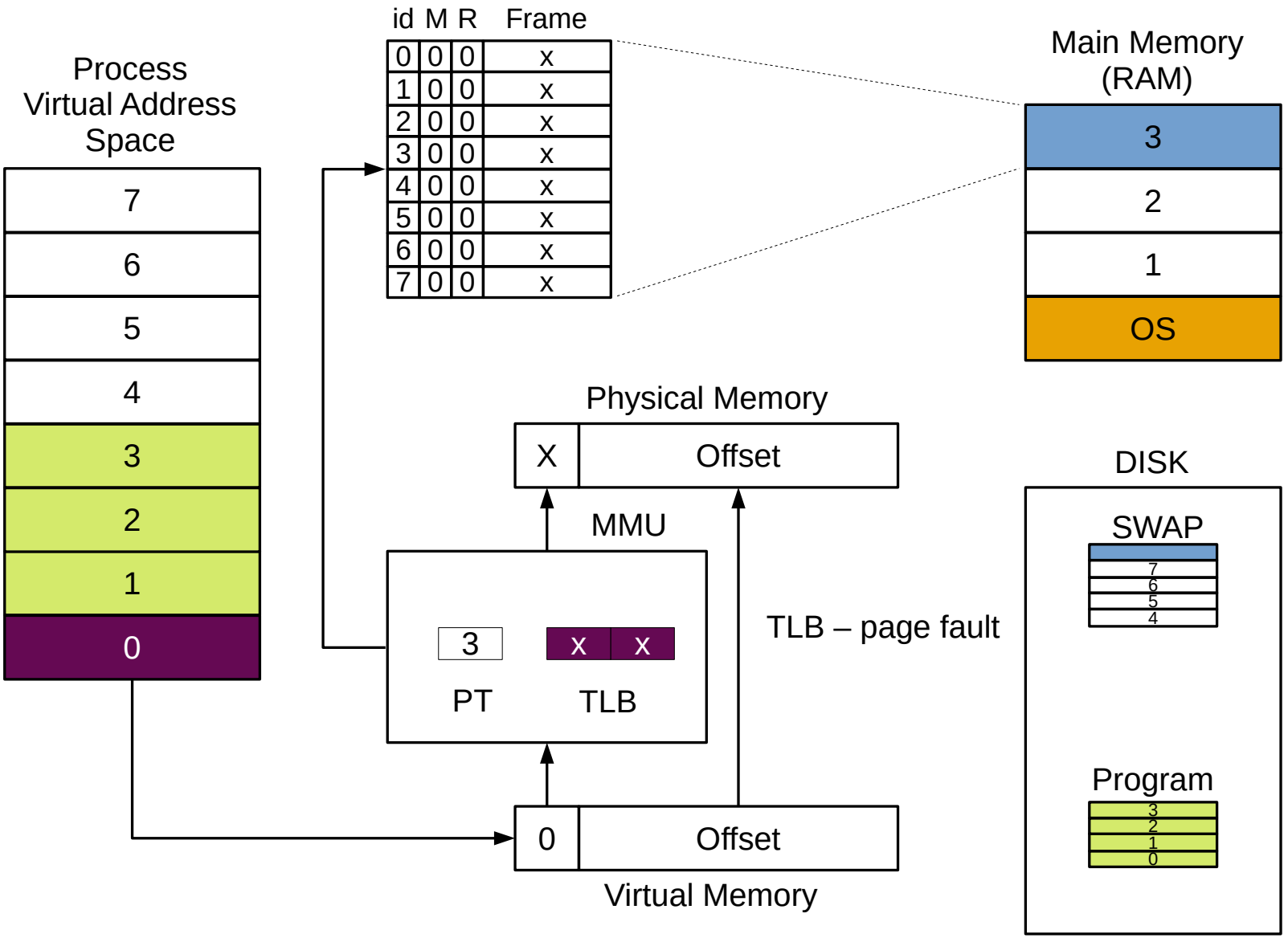
M-bit a R-bit

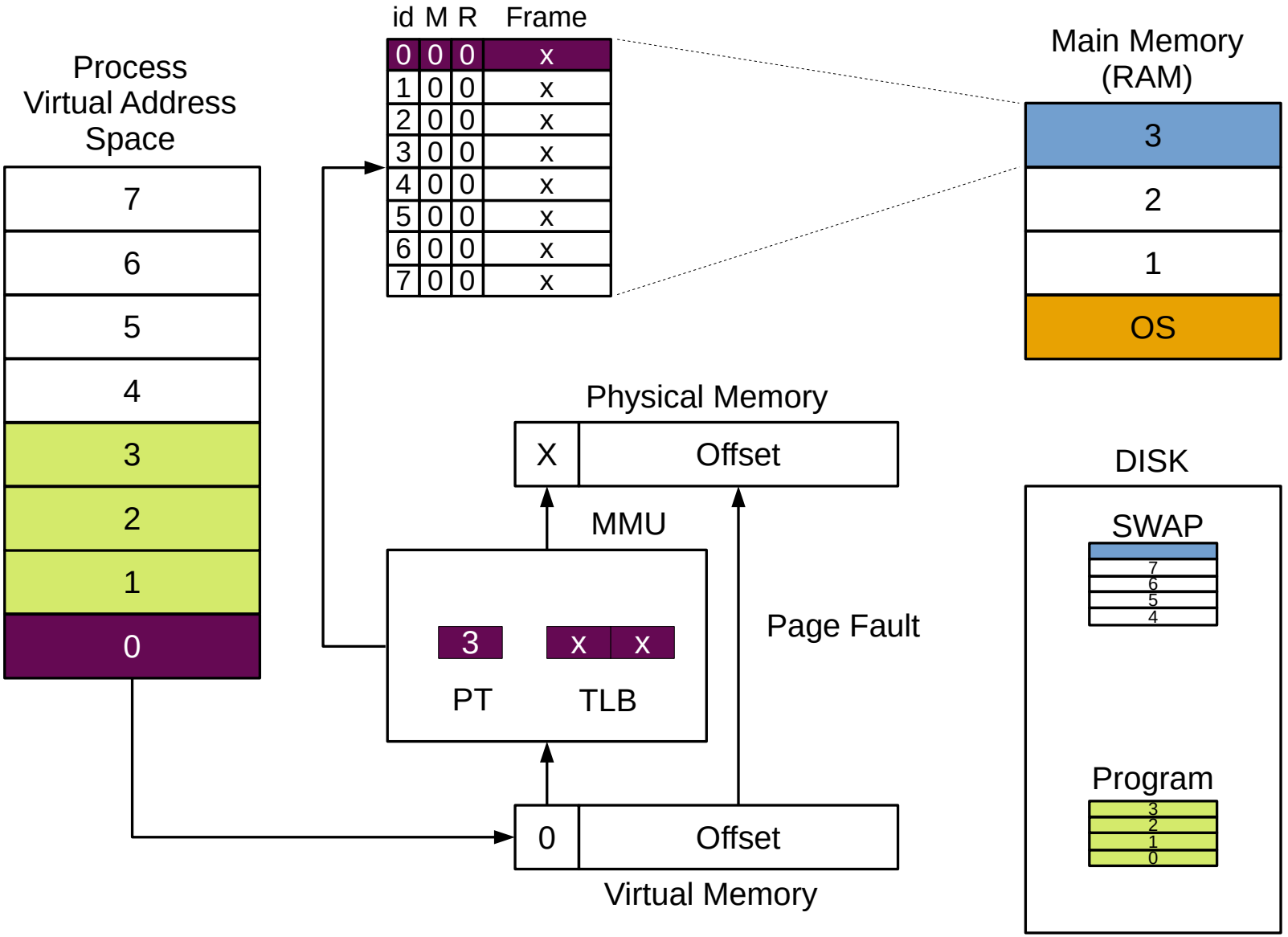
- Po spustení procesu všetky jeho stránky majú M a R = 0.
- Akonáhle je stránka použitá - R = 1.
- Akonáhle je stránka modifikovaná - M = 1.
- OS vykonáva reset R-bitu na 0:
  - Napríklad po každom hodinovom cykle.
  - Napríklad po každom Page Fault.
- OS vykonáva reset M-bitu až keď vykoná zápis stránky na disk.

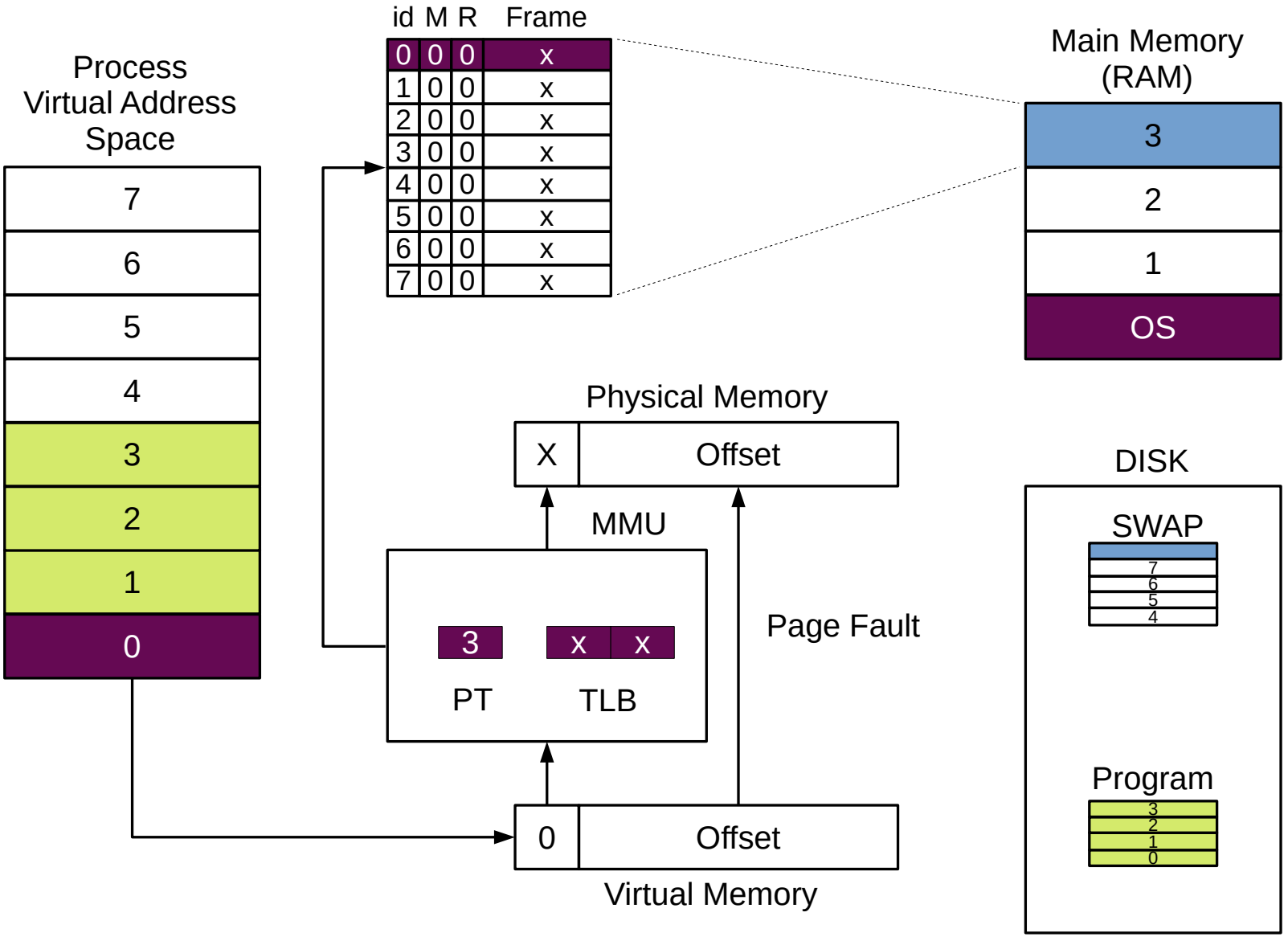

Process
Virtual Address
Space
7
6
5
4
3
2
1
0
Main Memory
(RAM)
3
2
1
OS
Physical Memory
X
Offset
MMU
x
x
x
PT
TLB
DISK
Program
3
2
1
0
X
Offset
Virtual Memory

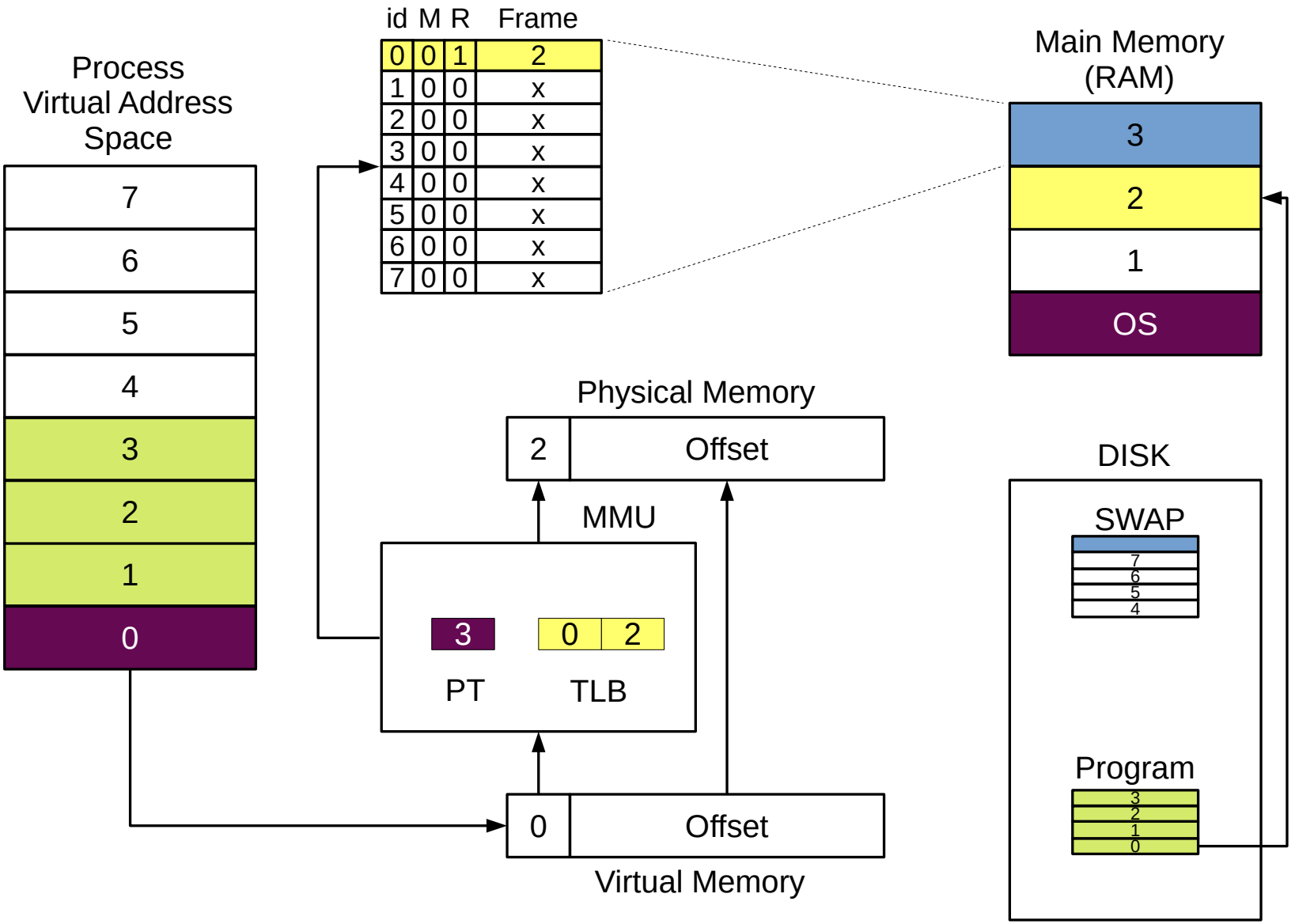

Process Virtual Address Space
7
6
5
4
3
2
1
0
id M R Frame
0 0 0 x
1 0 0 x
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 0 0 x
Main Memory (RAM)
3
2
1
OS
Physical Memory
X Offset
MMU
3
x x
PT
TLB
DISK
SWAP
7
6
5
4
Program
3
2
1
0
X Offset
Virtual Memory
Štart programu

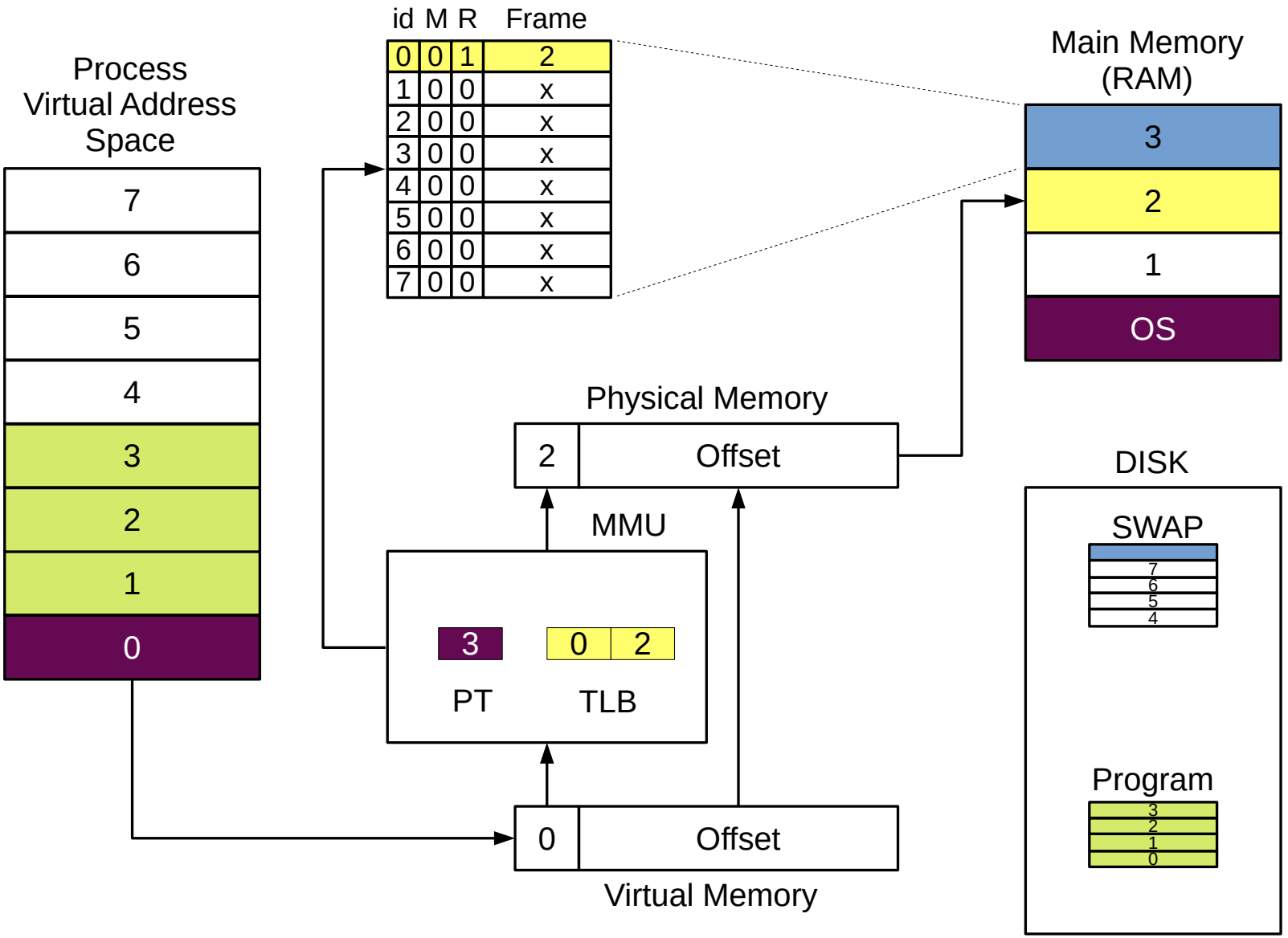

Process
Virtual Address
Space
7
6
5
4
3
2
1
0
id M R Frame
0 0 0 x
1 0 0 x
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 0 0 x
Main Memory
(RAM)
3
2
1
OS
Physical Memory
X
Offset
MMU
3
PT
x x
TLB
0
Offset
Virtual Memory
DISK
SWAP
7
6
5
4
Program
3
2
1
0
Čítanie prvých stránok

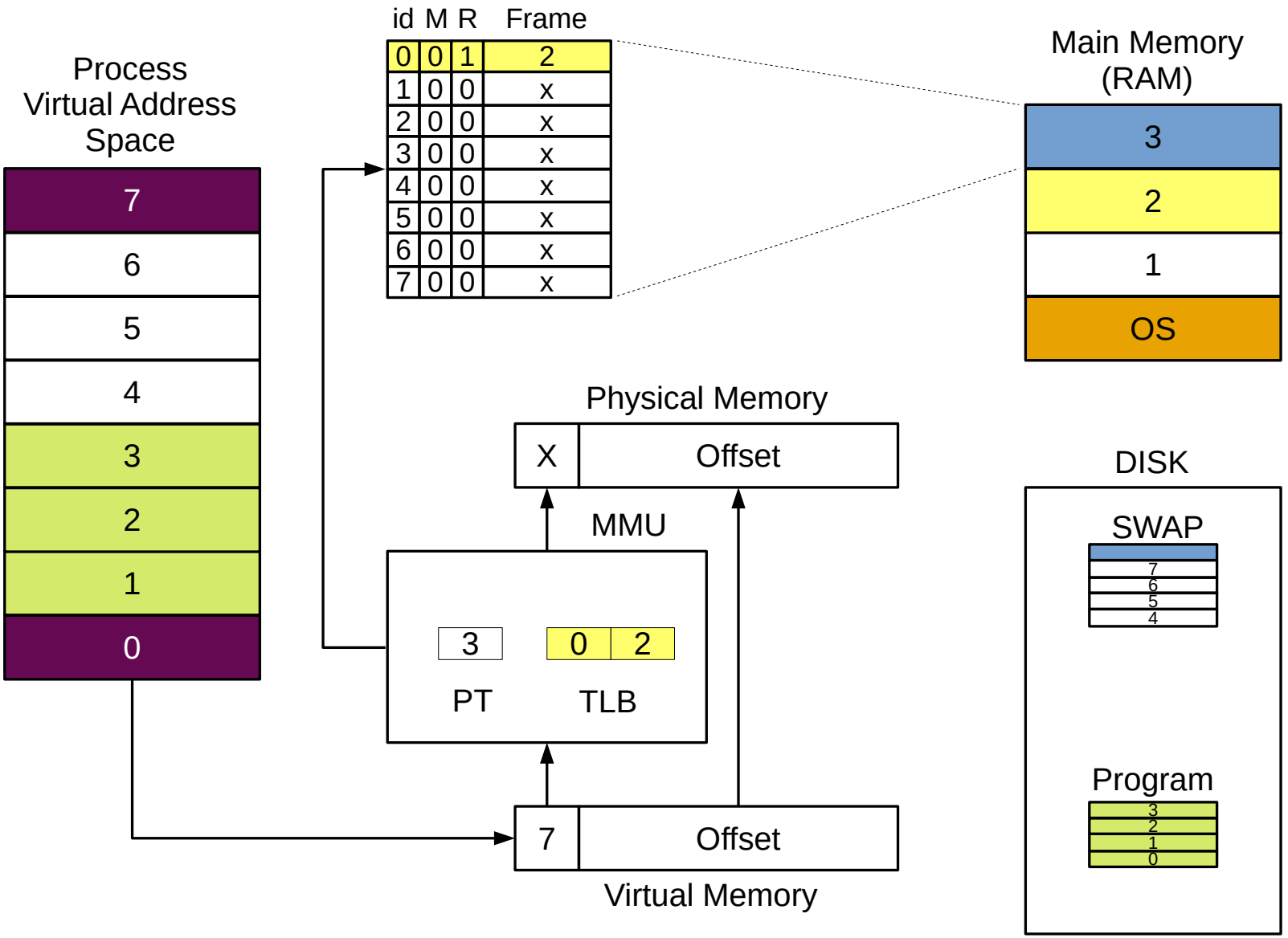

Process
Virtual Address
Space
7
6
5
4
3
2
1
0
id M R Frame
0 0 0 x
1 0 0 x
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 0 0 x
Main Memory
(RAM)
3
2
1
OS
Physical Memory
X
Offset
MMU
3
PT
x x
TLB
TLB – page fault
DISK
SWAP
7
6
5
4
Program
3
2
1
0
0
Offset
Virtual Memory

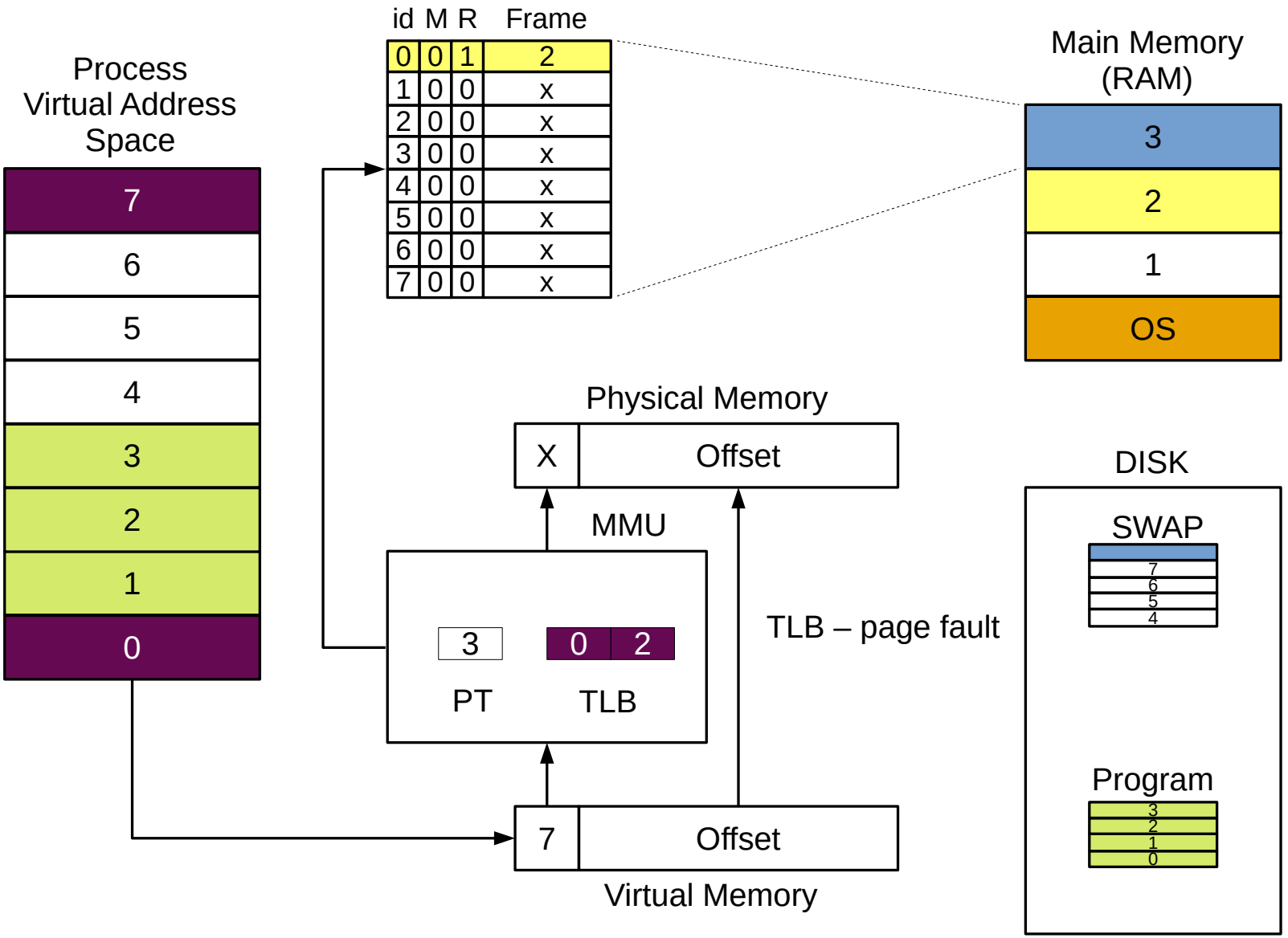

Process
Virtual Address
Space
7
6
5
4
3
2
1
0
id M R Frame
0 0 0 x
1 0 0 x
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 0 0 x
Main Memory
(RAM)
3
2
1
OS
Physical Memory
X
Offset
MMU
3
x x
PT
TLB
Page Fault
0
Offset
Virtual Memory
DISK
SWAP
7
6
5
4
Program
3
2
1
0

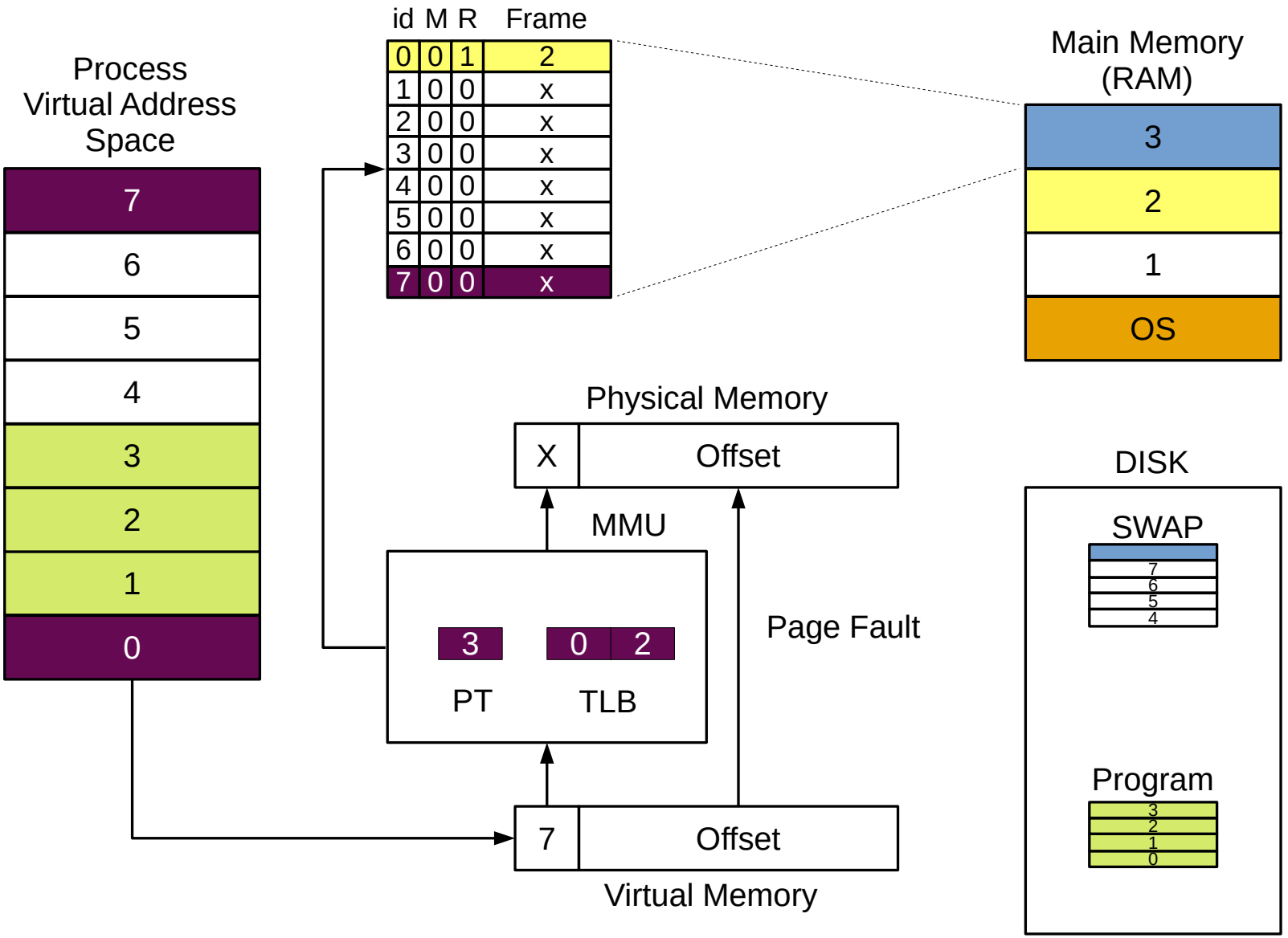

Process
Virtual Address
Space
7
6
5
4
3
2
1
0
id M R Frame
0 0 0 x
1 0 0 x
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 0 0 x
Main Memory
(RAM)
3
2
1
OS
Physical Memory
X
Offset
MMU
3
x x
PT
TLB
Page Fault
0
Offset
Virtual Memory
DISK
SWAP
7
6
5
4
Program
3
2
1
0

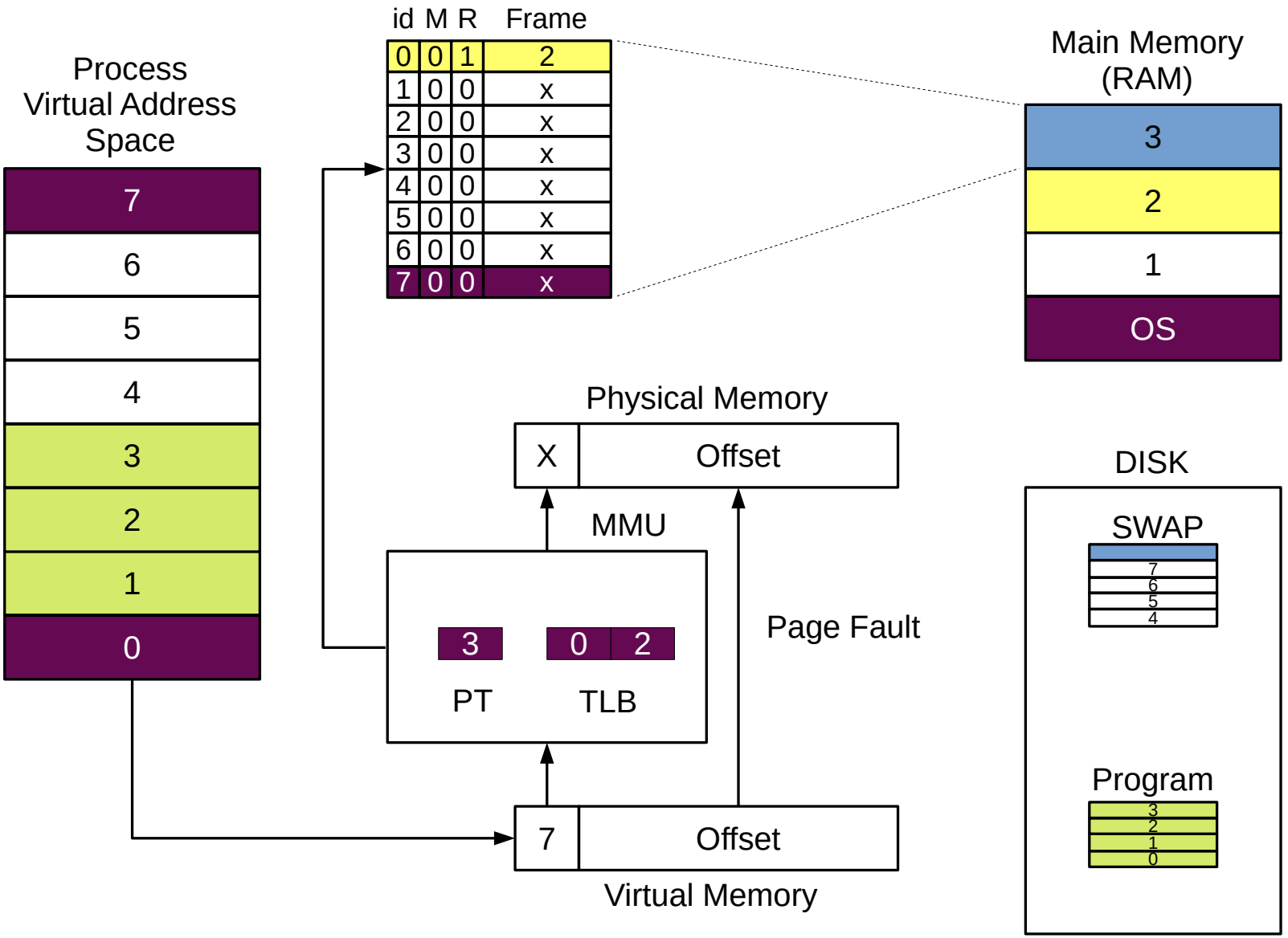

Process
Virtual Address
Space
7
6
5
4
3
2
1
0
id M R Frame
0 0 1 2
1 0 0 x
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 0 0 x
Main Memory
(RAM)
3
2
1
OS
Physical Memory
2 Offset
MMU
3
0 2
PT
TLB
DISK
SWAP
7
6
5
4
Program
3
2
1
0
0 Offset
Virtual Memory

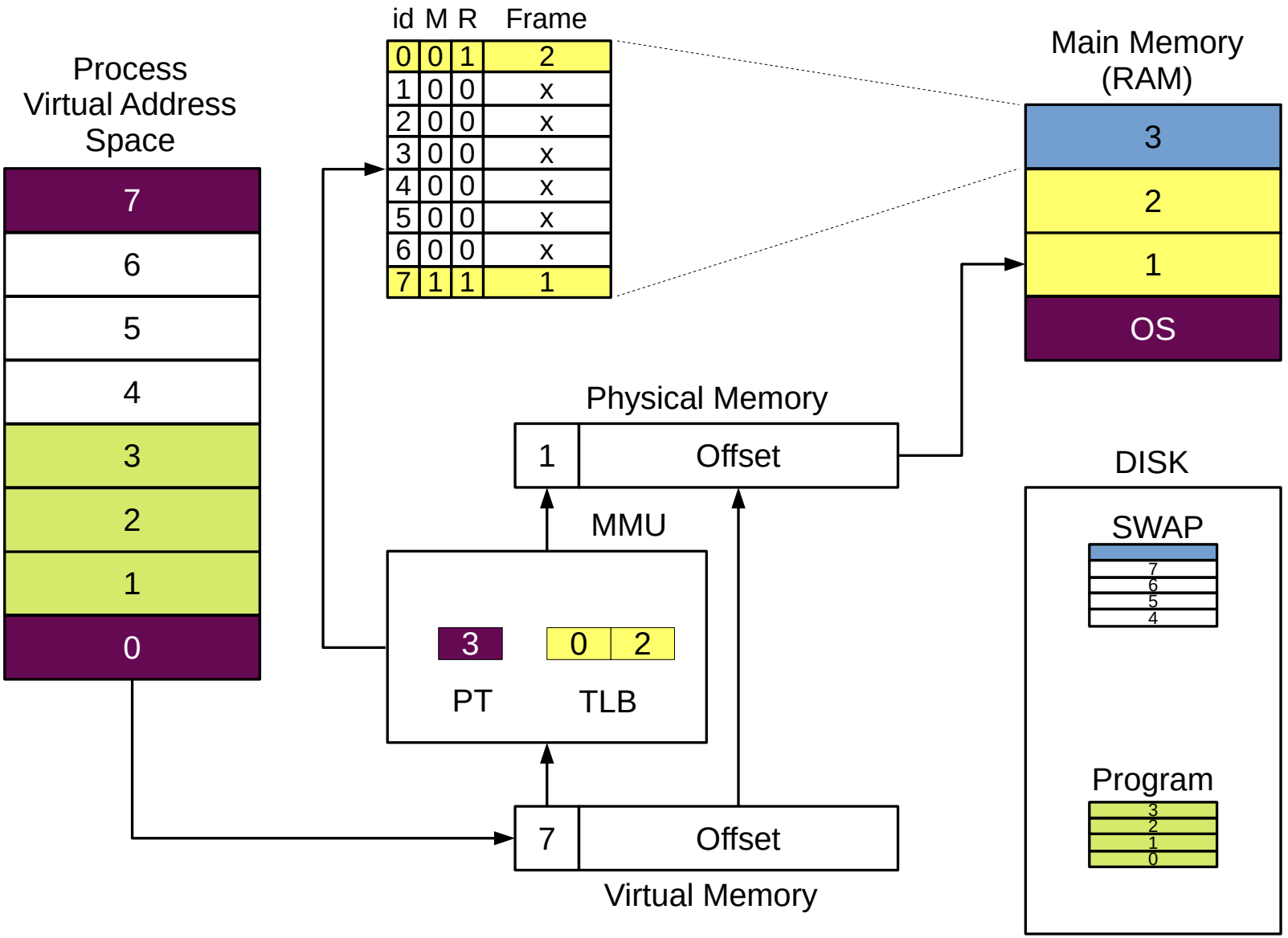

Process
Virtual Address
Space
7
6
5
4
3
2
1
0
id M R Frame
0 0 1 2
1 0 0 x
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 0 0 x
Main Memory
(RAM)
3
2
1
OS
Physical Memory
2 Offset
MMU
3
0 2
PT
TLB
0 Offset
Virtual Memory
DISK
SWAP
7
6
5
4
Program
3
2
1
0

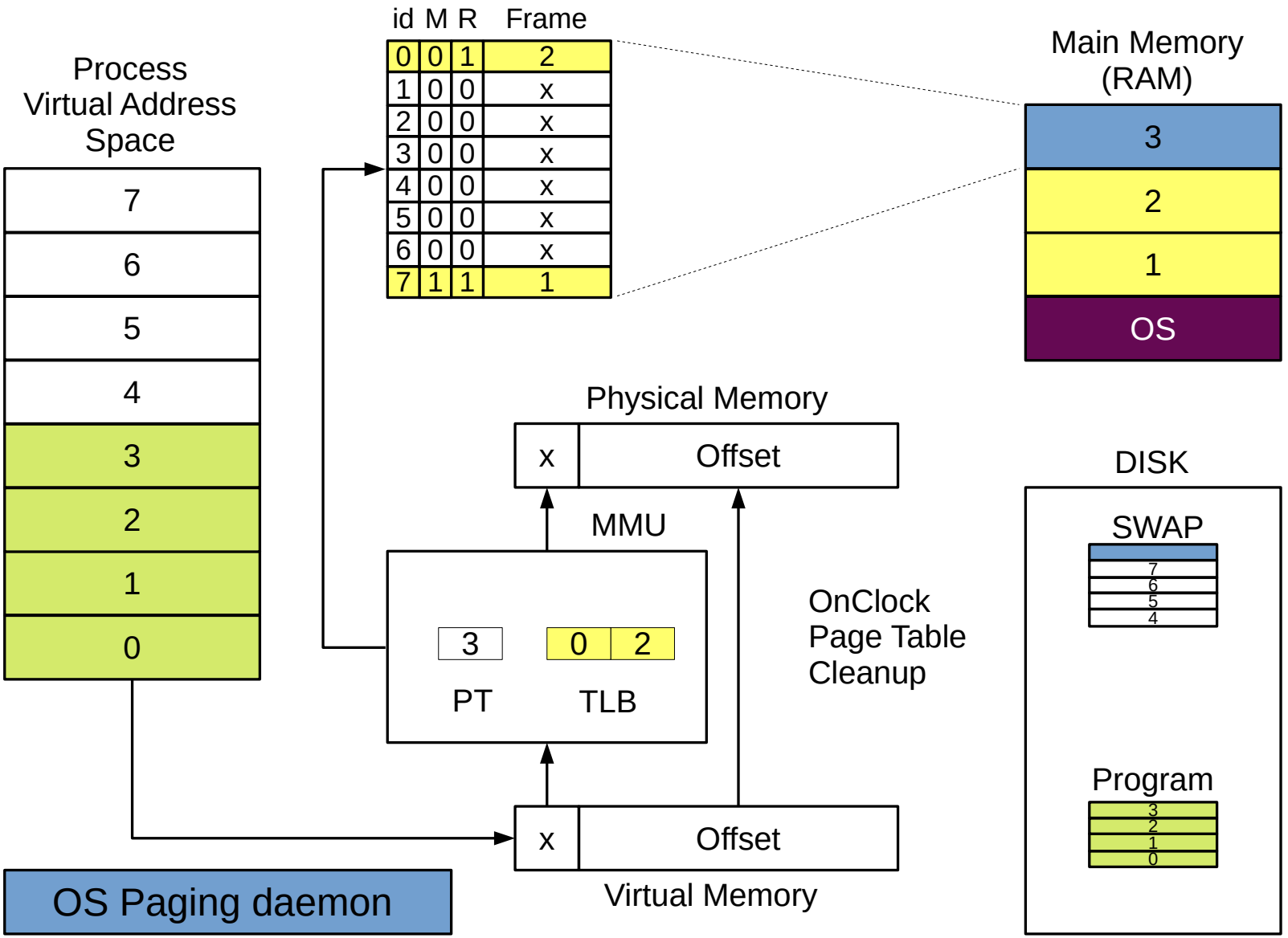

Process
Virtual Address
Space
7
6
5
4
3
2
1
0
id M R Frame
0 0 1 2
1 0 0 x
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 0 0 x
Main Memory
(RAM)
3
2
1
OS
Physical Memory
X
Offset
MMU
3
0 2
PT
TLB
7
Offset
Virtual Memory
DISK
SWAP
7
6
5
4
Program
3
2
1
0

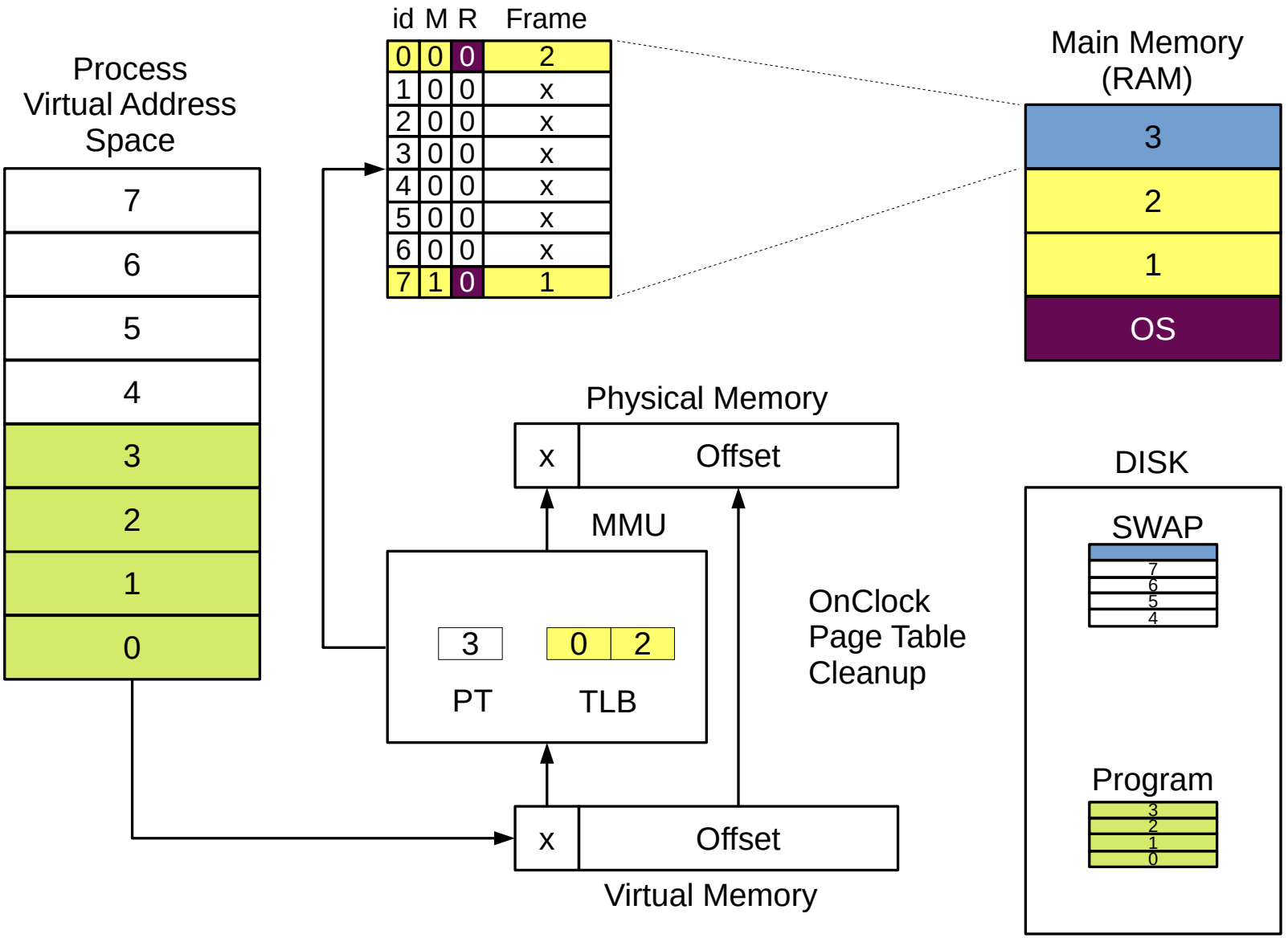

Process
Virtual Address
Space
7
6
5
4
3
2
1
0
id M R Frame
0 0 1 2
1 0 0 x
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 0 0 x
Main Memory
(RAM)
3
2
1
OS
Physical Memory
X
Offset
MMU
3
PT
0 2
TLB
TLB – page fault
DISK
SWAP
7
6
5
4
Program
3
2
1
0
7
Offset
Virtual Memory

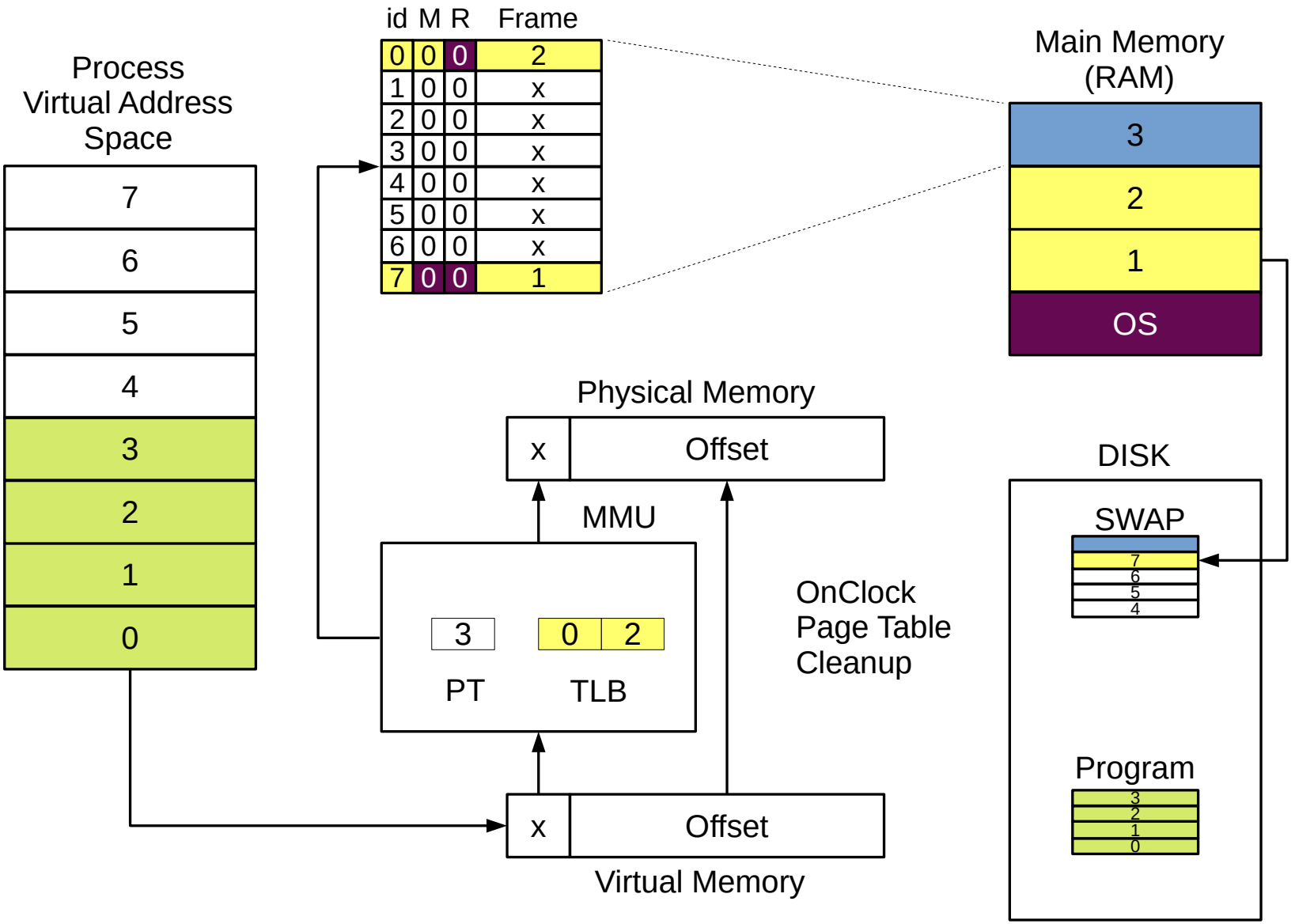

Process
Virtual Address
Space
7
6
5
4
3
2
1
0
id M R Frame
0 0 1 2
1 0 0 x
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 0 0 x
Main Memory
(RAM)
3
2
1
OS
Physical Memory
X
Offset
MMU
3
0 2
PT
TLB
Page Fault
DISK
SWAP
7
6
5
4
Program
3
2
1
0
7
Offset
Virtual Memory

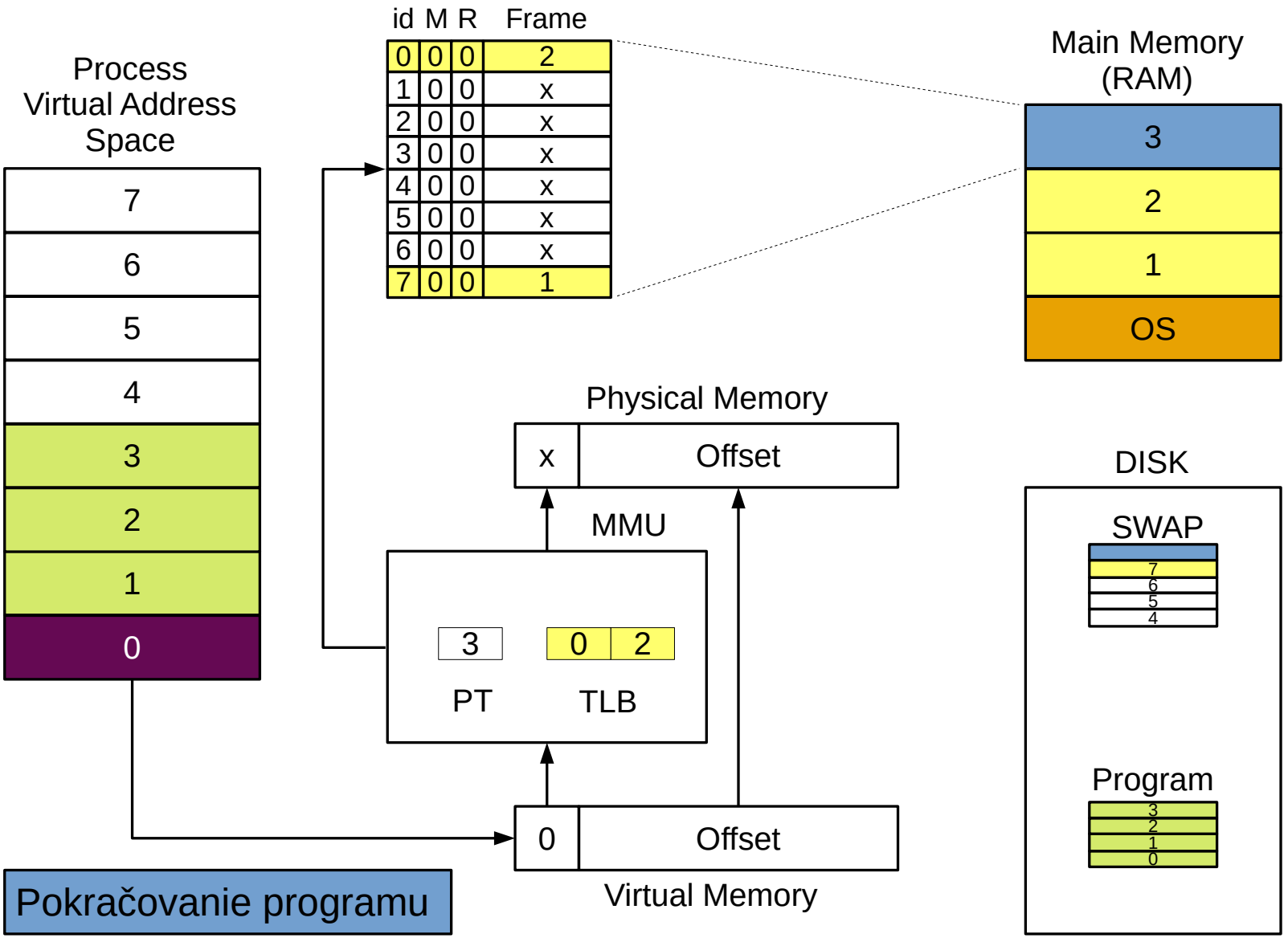

Process
Virtual Address
Space
7
6
5
4
3
2
1
0
id M R Frame
0 0 1 2
1 0 0 x
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 0 0 x
Main Memory
(RAM)
3
2
1
OS
Physical Memory
X
Offset
MMU
3
0 2
PT
TLB
Page Fault
DISK
SWAP
7
6
5
4
Program
3
2
1
0
7
Offset
Virtual Memory

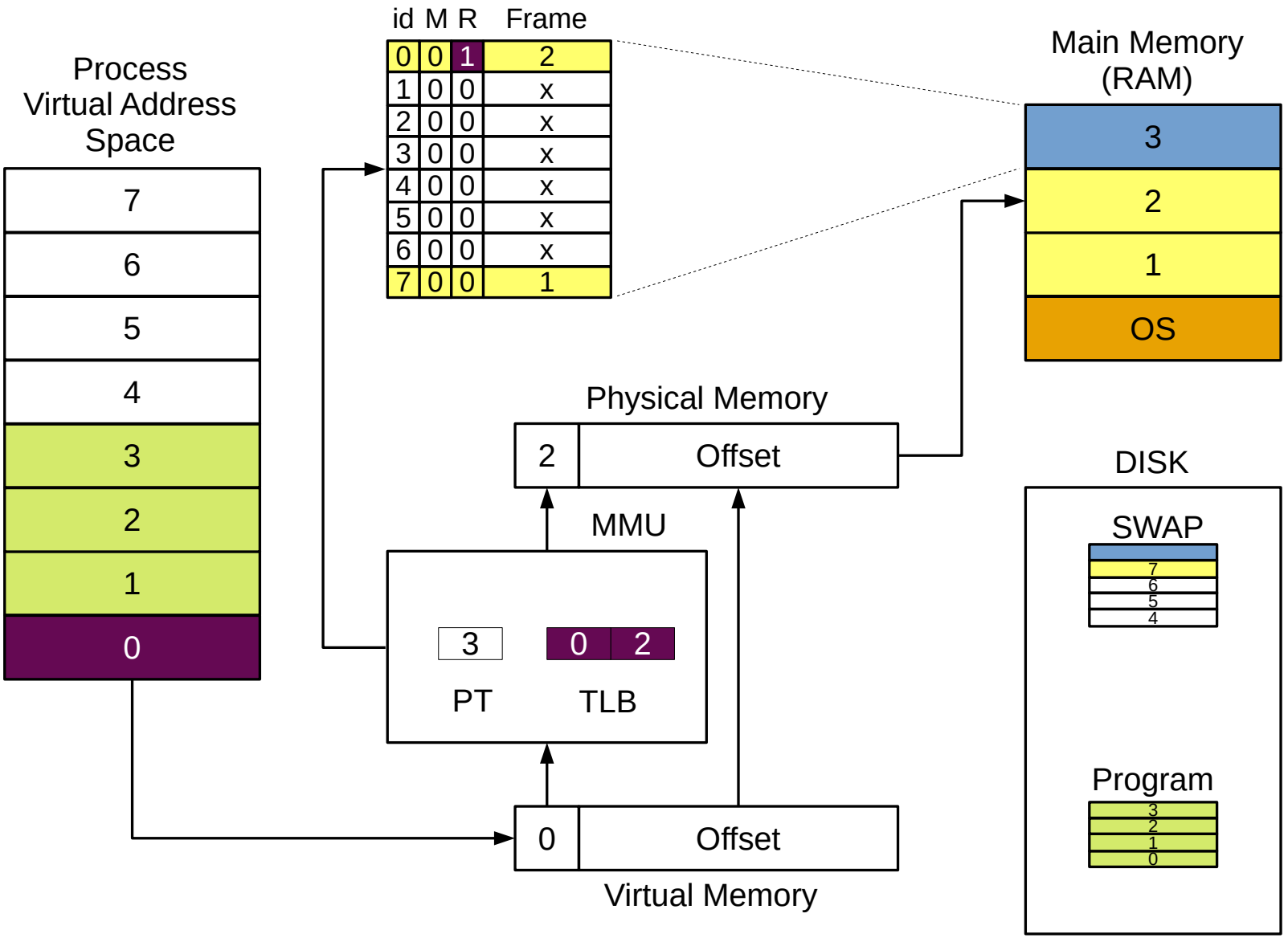

Process
Virtual Address
Space
7
6
5
4
3
2
1
0
id M R Frame
0 0 1 2
1 0 0 x
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 1 1 1
Main Memory
(RAM)
3
2
1
OS
Physical Memory
1
Offset
MMU
3
0 2
PT
TLB
7
Offset
Virtual Memory
DISK
SWAP
7
6
5
4
Program
3
2
1
0

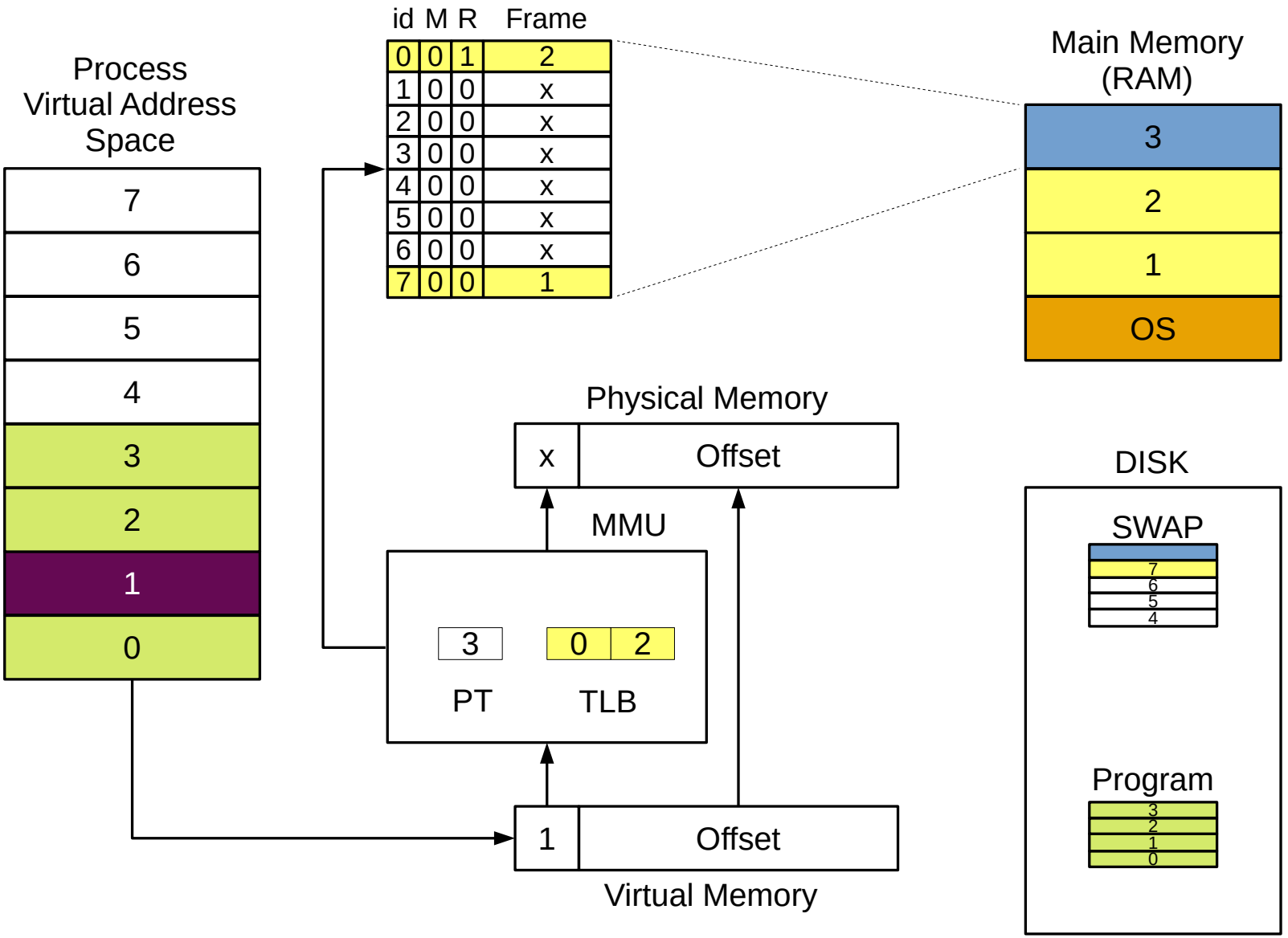

id M R Frame
0 0 1 2
1 0 0 x
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 1 1 1
Process Virtual Address Space
7
6
5
4
3
2
1
0
Main Memory (RAM)
3
2
1
OS
Physical Memory
x
Offset
MMU
3
0 2
PT
TLB
OnClock Page Table Cleanup
DISK
SWAP
7
6
5
4
Program
3
2
1
0
x
Offset
Virtual Memory
OS Paging daemon

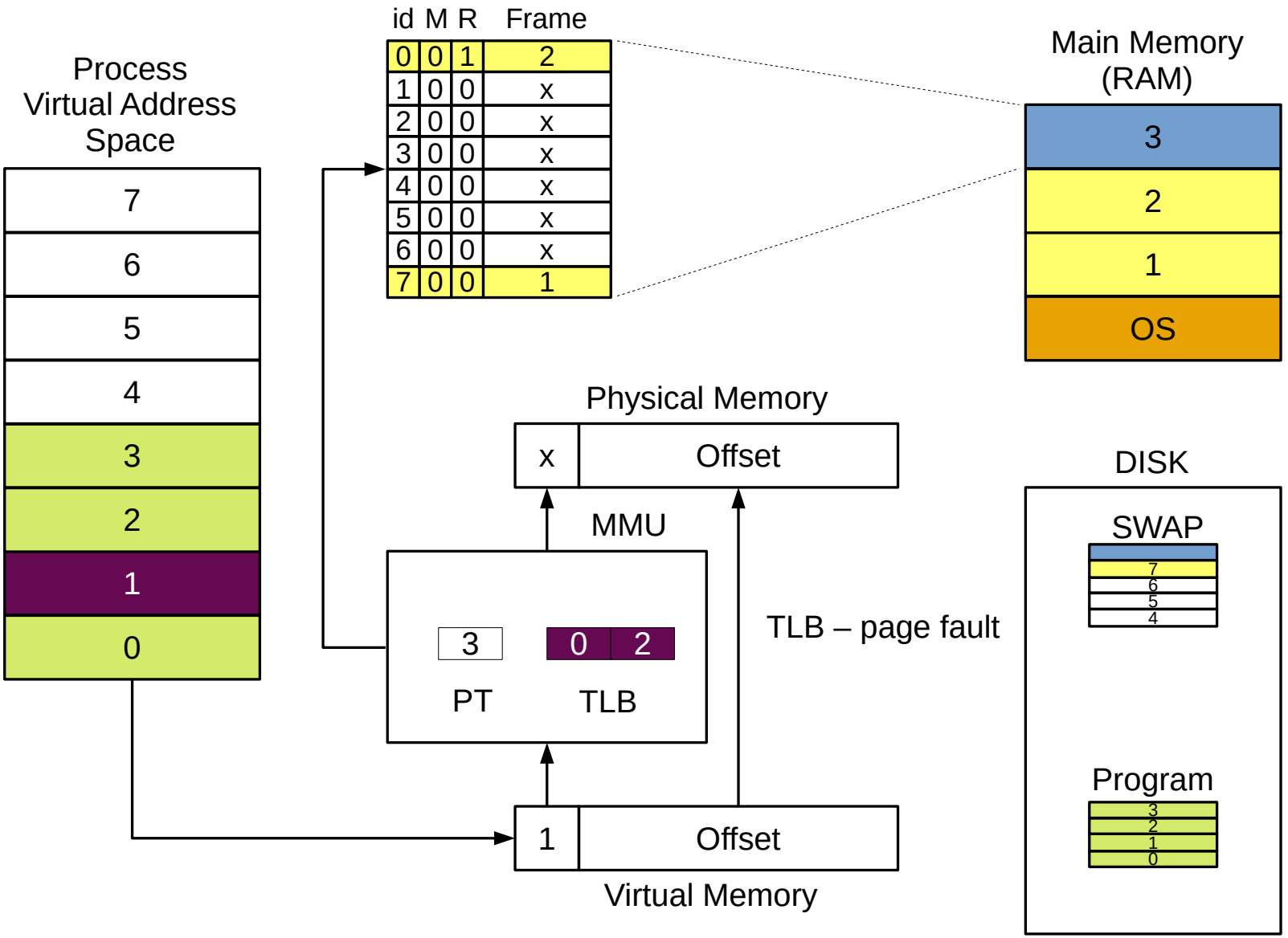

Process
Virtual Address
Space
7
6
5
4
3
2
1
0
id M R Frame
0 0 0 2
1 0 0 x
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 1 0 1
Main Memory
(RAM)
3
2
1
OS
Physical Memory
x Offset
MMU
3
0 2
PT
TLB
OnClock
Page Table
Cleanup
DISK
SWAP
7
6
5
4
Program
3
2
1
0
x Offset
Virtual Memory

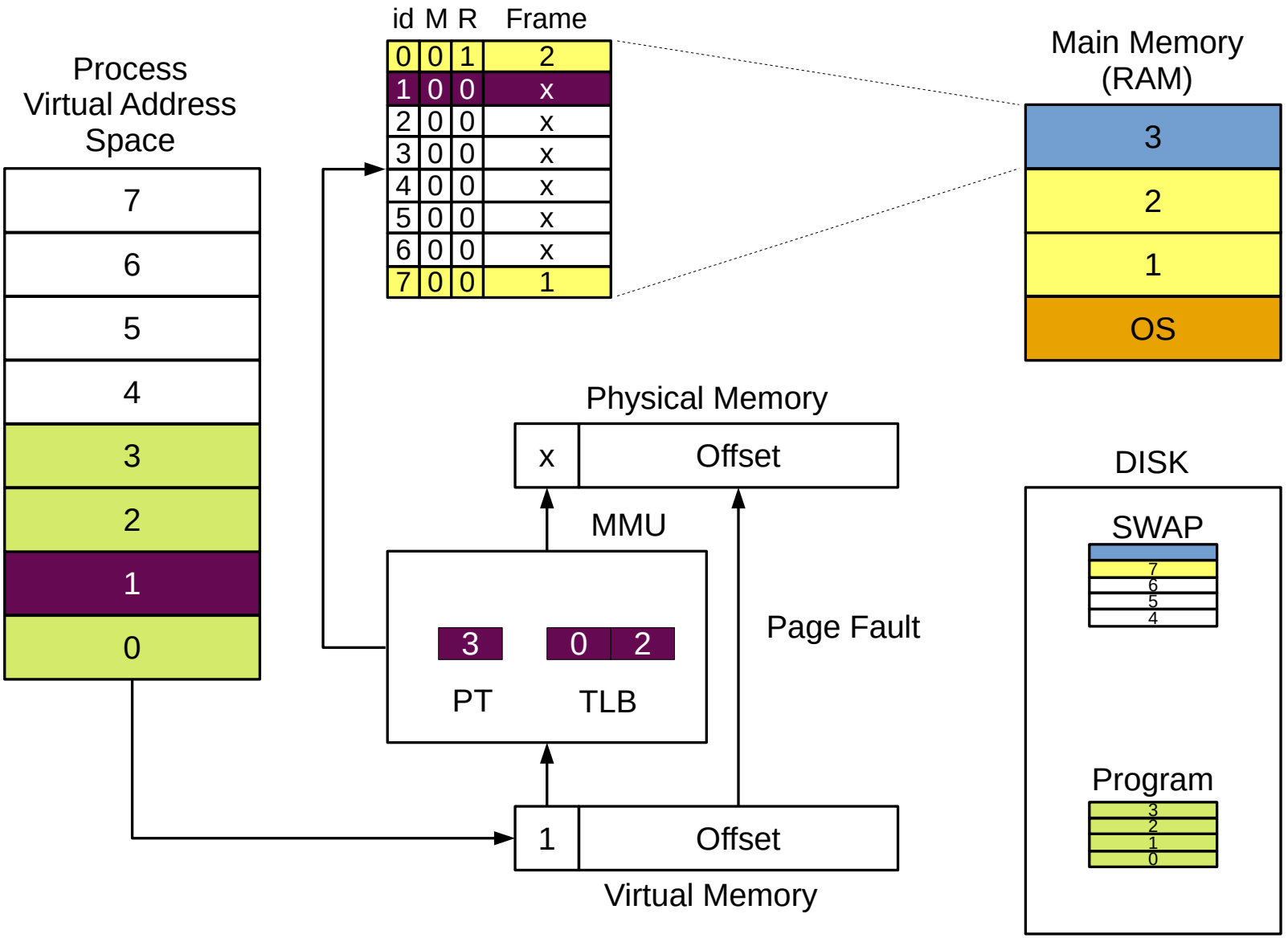

Process
Virtual Address
Space
7
6
5
4
3
2
1
0
id M R Frame
0 0 0 2
1 0 0 x
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 0 0 1
Main Memory
(RAM)
3
2
1
OS
Physical Memory
x Offset
MMU
3
0 2
PT
TLB
OnClock
Page Table
Cleanup
DISK
SWAP
7
6
5
4
Program
3
2
1
0
x Offset
Virtual Memory

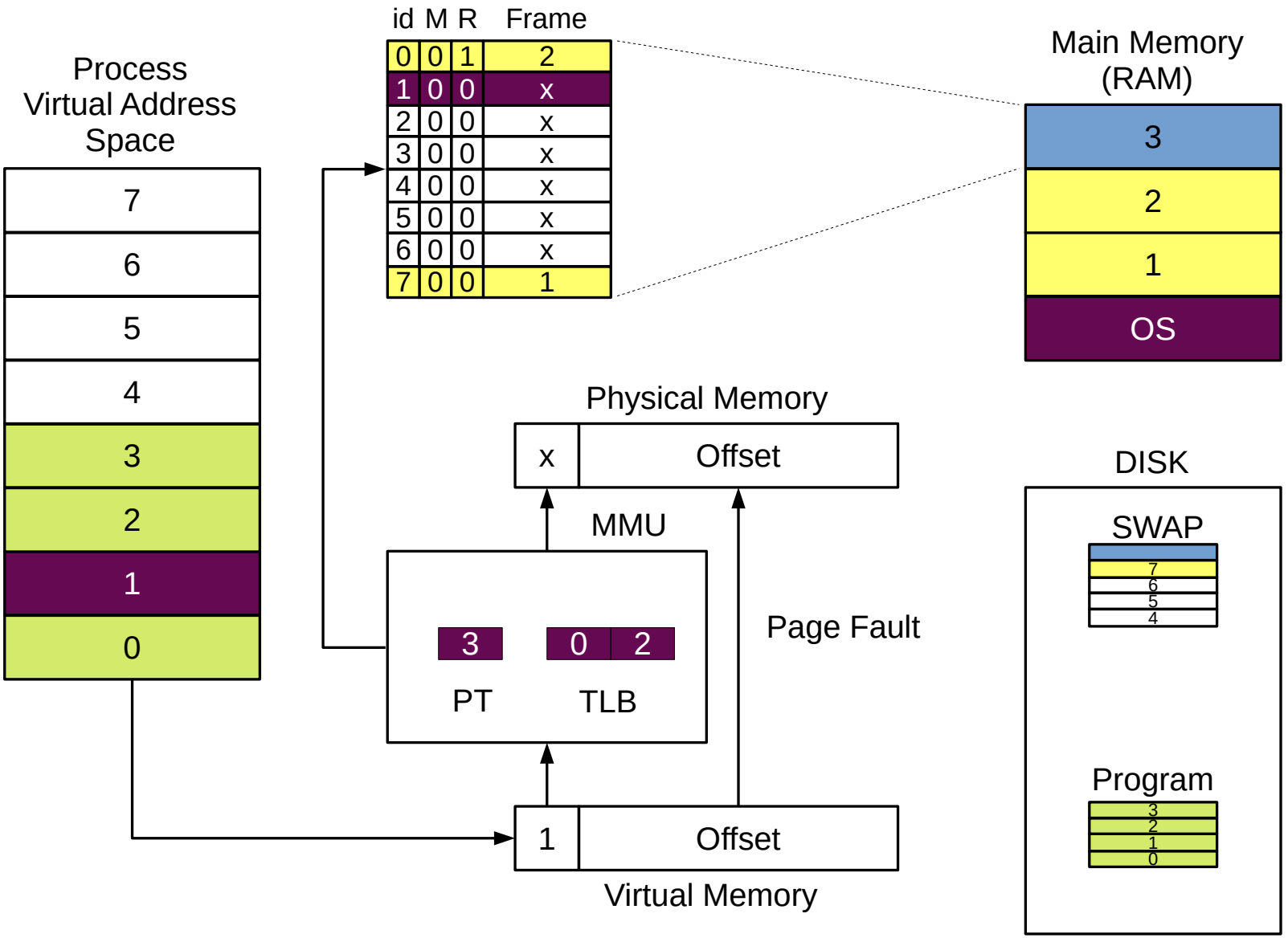

Process
Virtual Address
Space
7
6
5
4
3
2
1
0
id M R Frame
0 0 0 2
1 0 0 x
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 0 0 1
Main Memory
(RAM)
3
2
1
OS
Physical Memory
x
Offset
MMU
3
0 2
PT
TLB
DISK
SWAP
7
6
5
4
Program
3
2
1
0
0
Offset
Virtual Memory
Pokračovanie programu

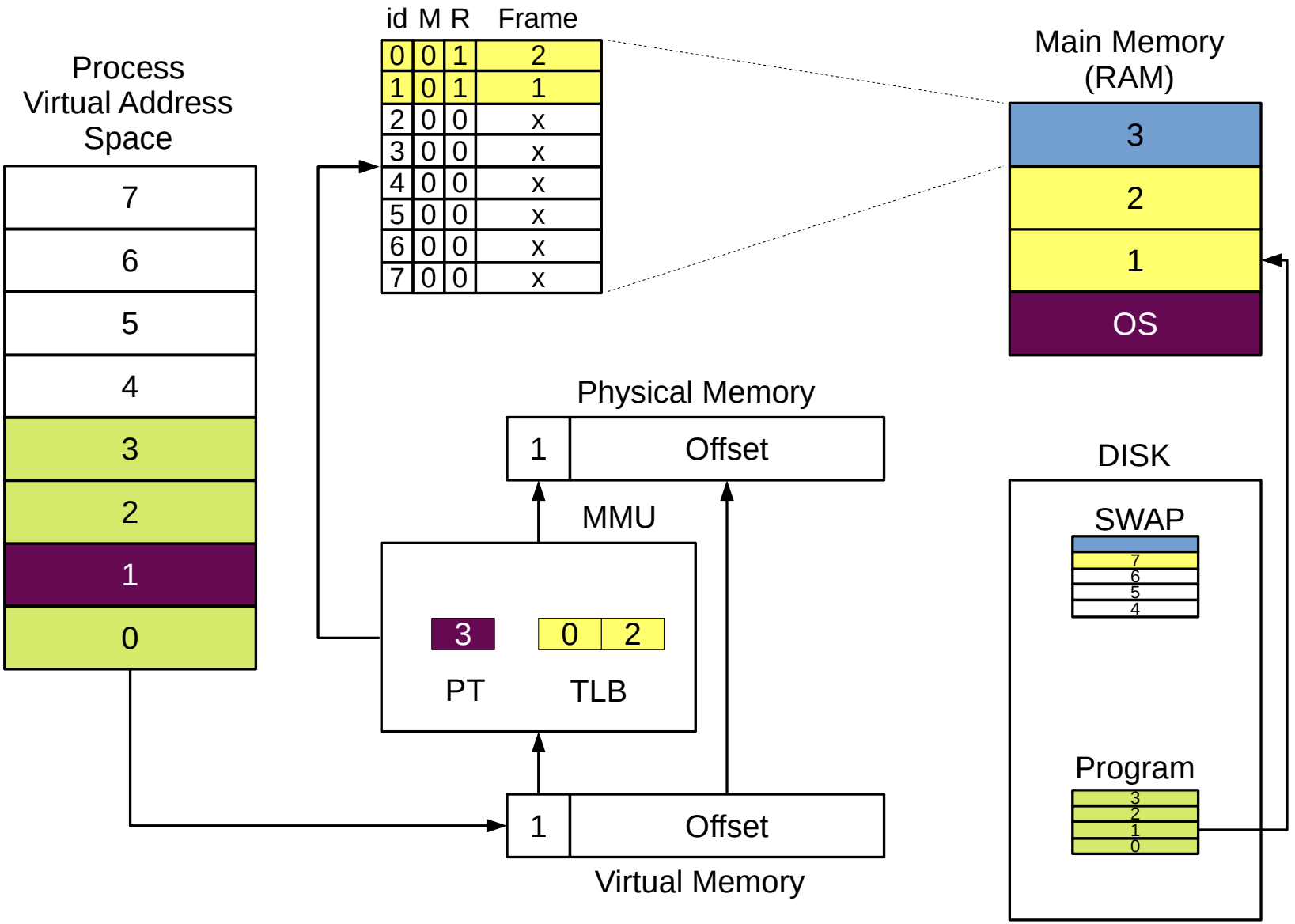

Process
Virtual Address
Space
7
6
5
4
3
2
1
0
id M R Frame
0 0 1 2
1 0 0 x
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 0 0 1
Main Memory
(RAM)
3
2
1
OS
Physical Memory
2 Offset
MMU
3
PT
0 2
TLB
DISK
SWAP
7
6
5
4
Program
3
2
1
0
0 Offset
Virtual Memory

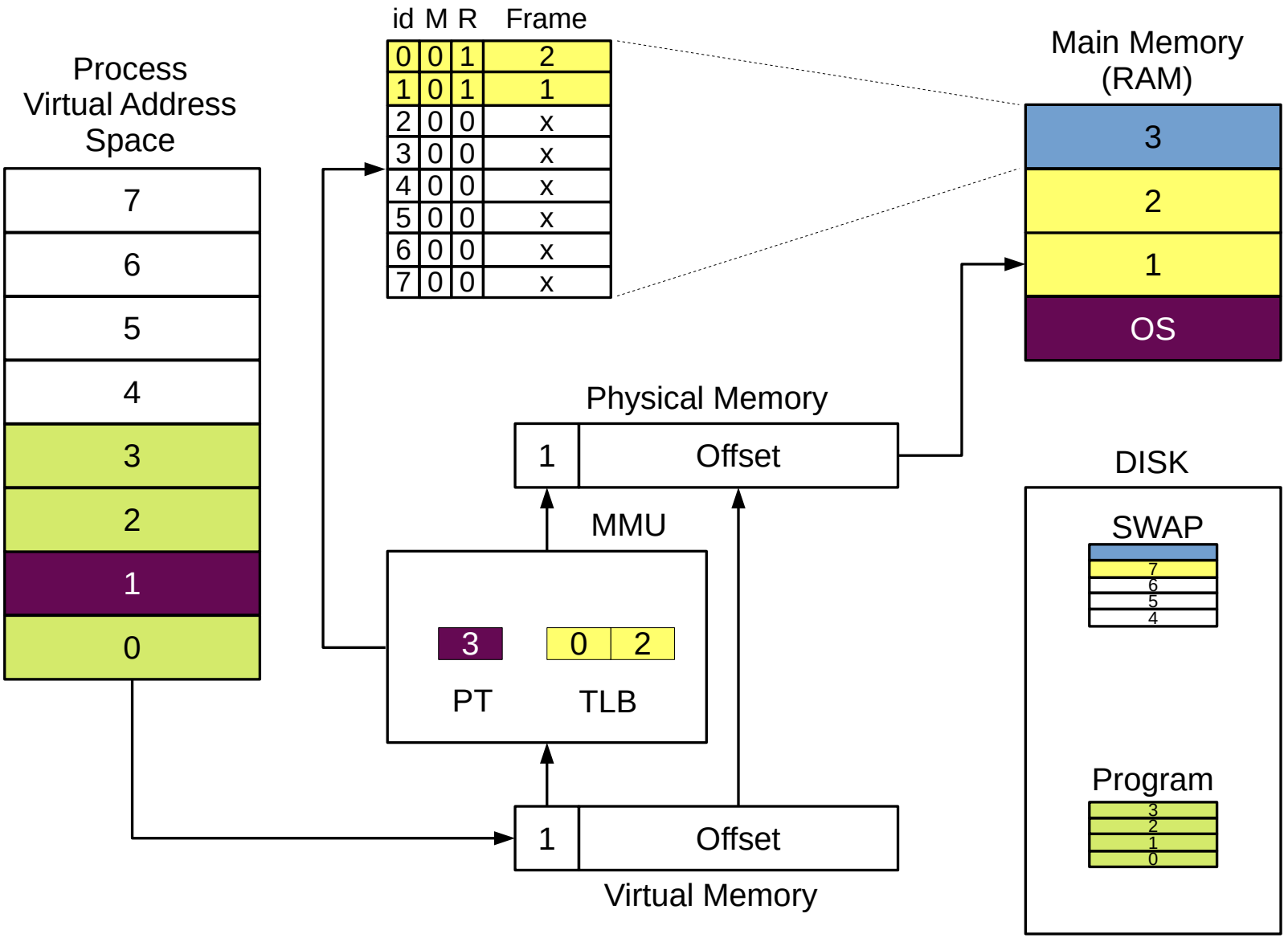

Process
Virtual Address
Space
7
6
5
4
3
2
1
0
id M R Frame
0 0 1 2
1 0 0 x
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 0 0 1
Main Memory
(RAM)
3
2
1
OS
Physical Memory
x
Offset
MMU
3
0 2
PT
TLB
1
Offset
Virtual Memory
DISK
SWAP
7
6
5
4
Program
3
2
1
0

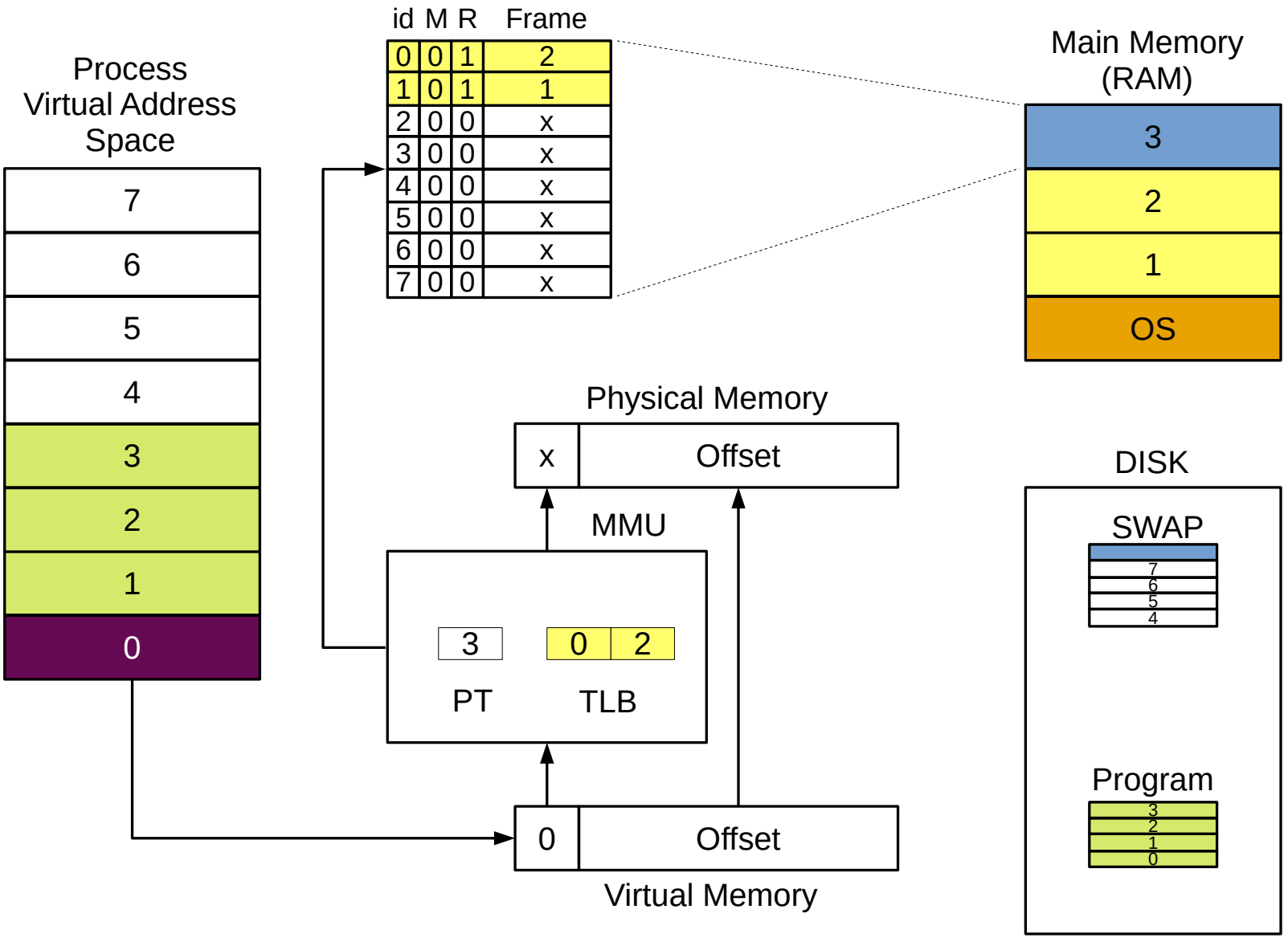

Process
Virtual Address
Space
7
6
5
4
3
2
1
0
id M R Frame
0 0 1 2
1 0 0 x
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 0 0 1
Main Memory
(RAM)
3
2
1
OS
Physical Memory
x
Offset
MMU
3
PT
0 2
TLB
TLB – page fault
DISK
SWAP
7
6
5
4
Program
3
2
1
0
1
Offset
Virtual Memory

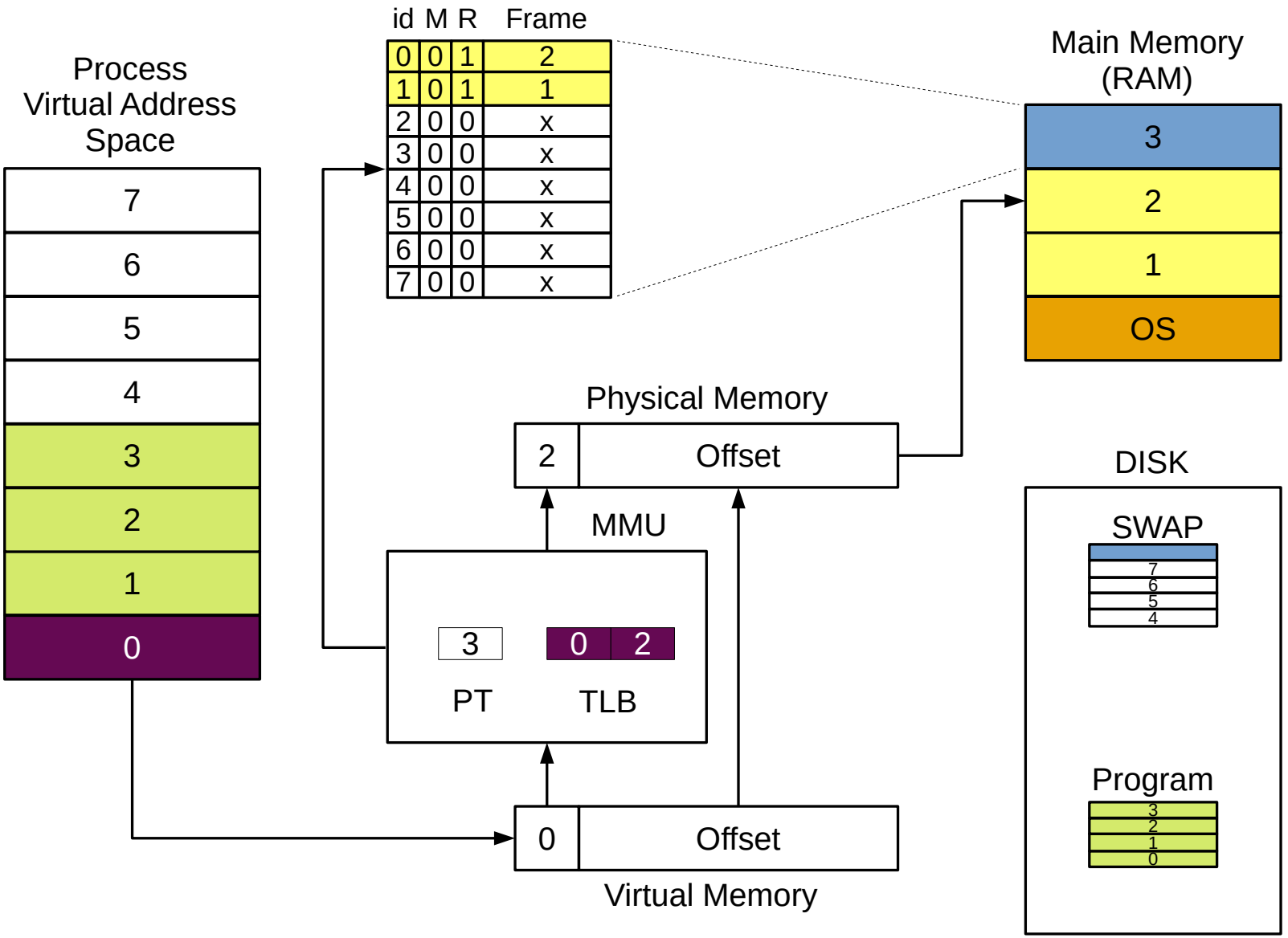

Process
Virtual Address
Space
7
6
5
4
3
2
1
0
id M R Frame
0 0 1 2
1 0 0 x
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 0 0 1
Main Memory
(RAM)
3
2
1
OS
Physical Memory
x Offset
MMU
3
PT
0 2
TLB
Page Fault
DISK
SWAP
7
6
5
4
Program
3
2
1
0
1 Offset
Virtual Memory

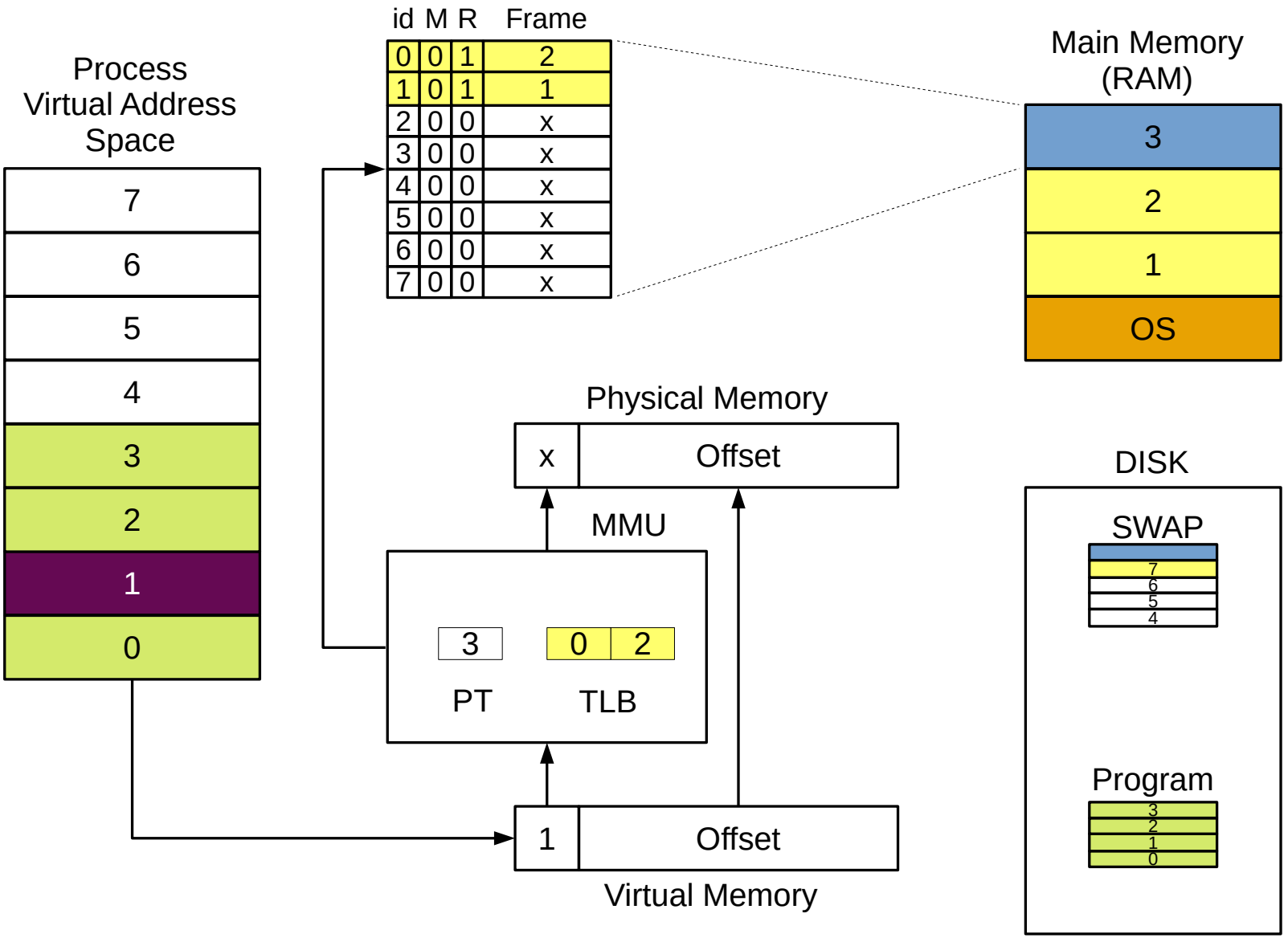

Process
Virtual Address
Space
7
6
5
4
3
2
1
0
id M R Frame
0 0 1 2
1 0 0 x
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 0 0 1
Main Memory
(RAM)
3
2
1
OS
Physical Memory
x Offset
MMU
3
PT
0 2
TLB
Page Fault
DISK
SWAP
7
6
5
4
Program
3
2
1
0
1 Offset
Virtual Memory

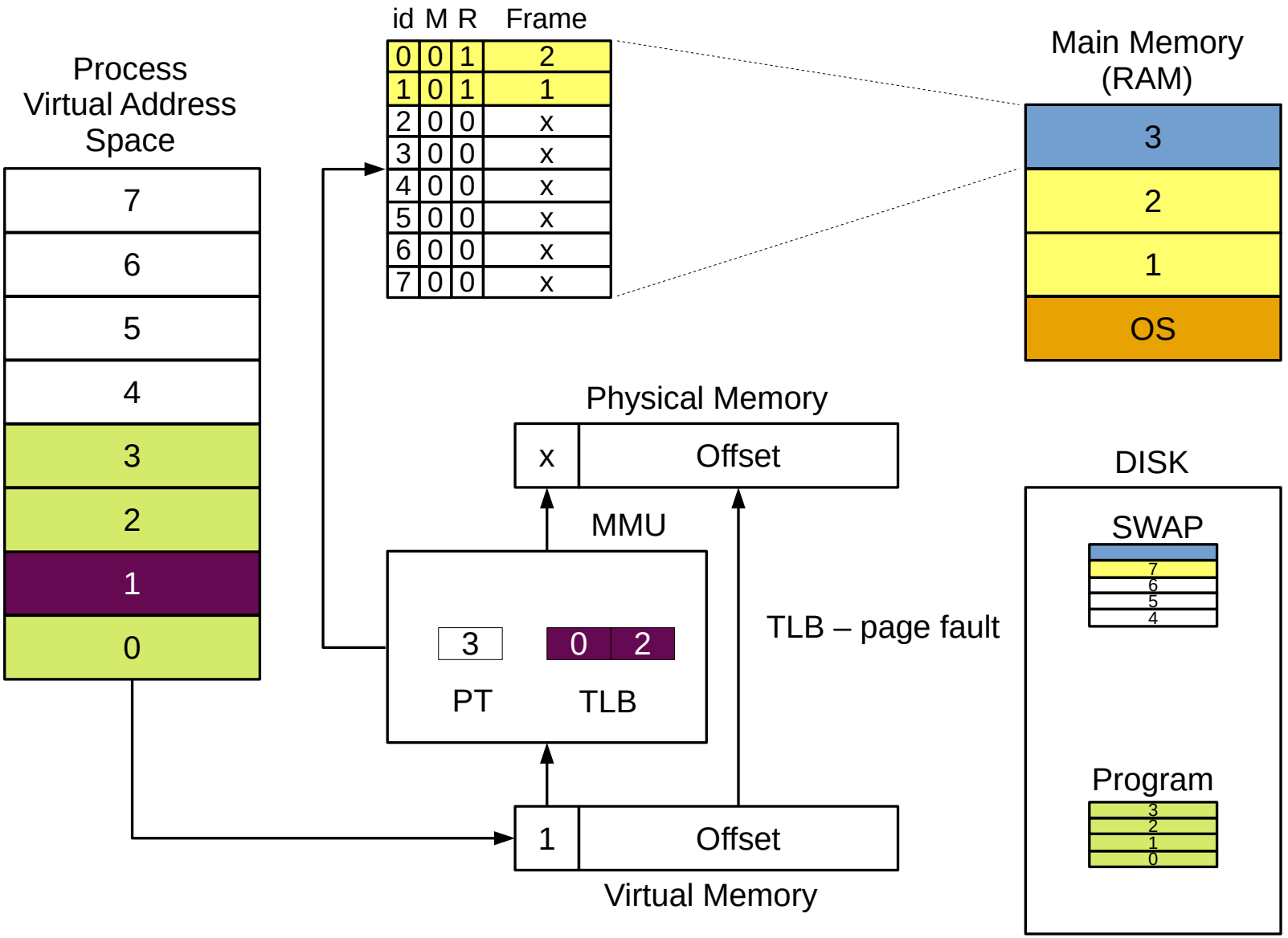

Process
Virtual Address
Space
7
6
5
4
3
2
1
0
id M R Frame
0 0 1 2
1 0 1 1
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 0 0 x
Main Memory
(RAM)
3
2
1
OS
Physical Memory
1 Offset
MMU
3
0 2
PT
TLB
1 Offset
Virtual Memory
DISK
SWAP
7
6
5
4
Program
3
2
1
0

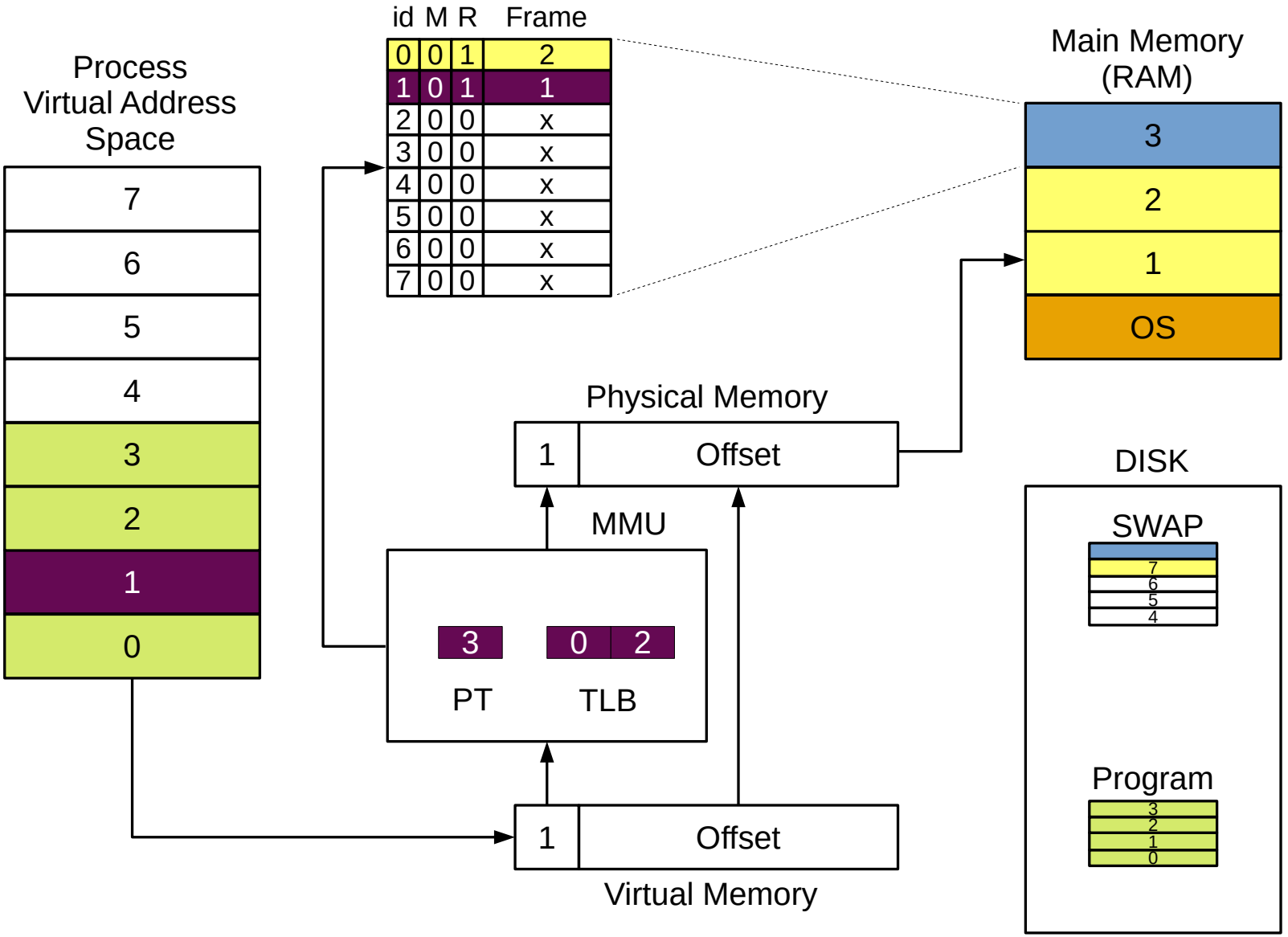

Process
Virtual Address
Space
7
6
5
4
3
2
1
0
id M R Frame
0 0 1 2
1 0 1 1
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 0 0 x
Main Memory
(RAM)
3
2
1
OS
Physical Memory
1 Offset
MMU
3
PT
0 2
TLB
DISK
SWAP
7
6
5
4
Program
3
2
1
0
1 Offset
Virtual Memory

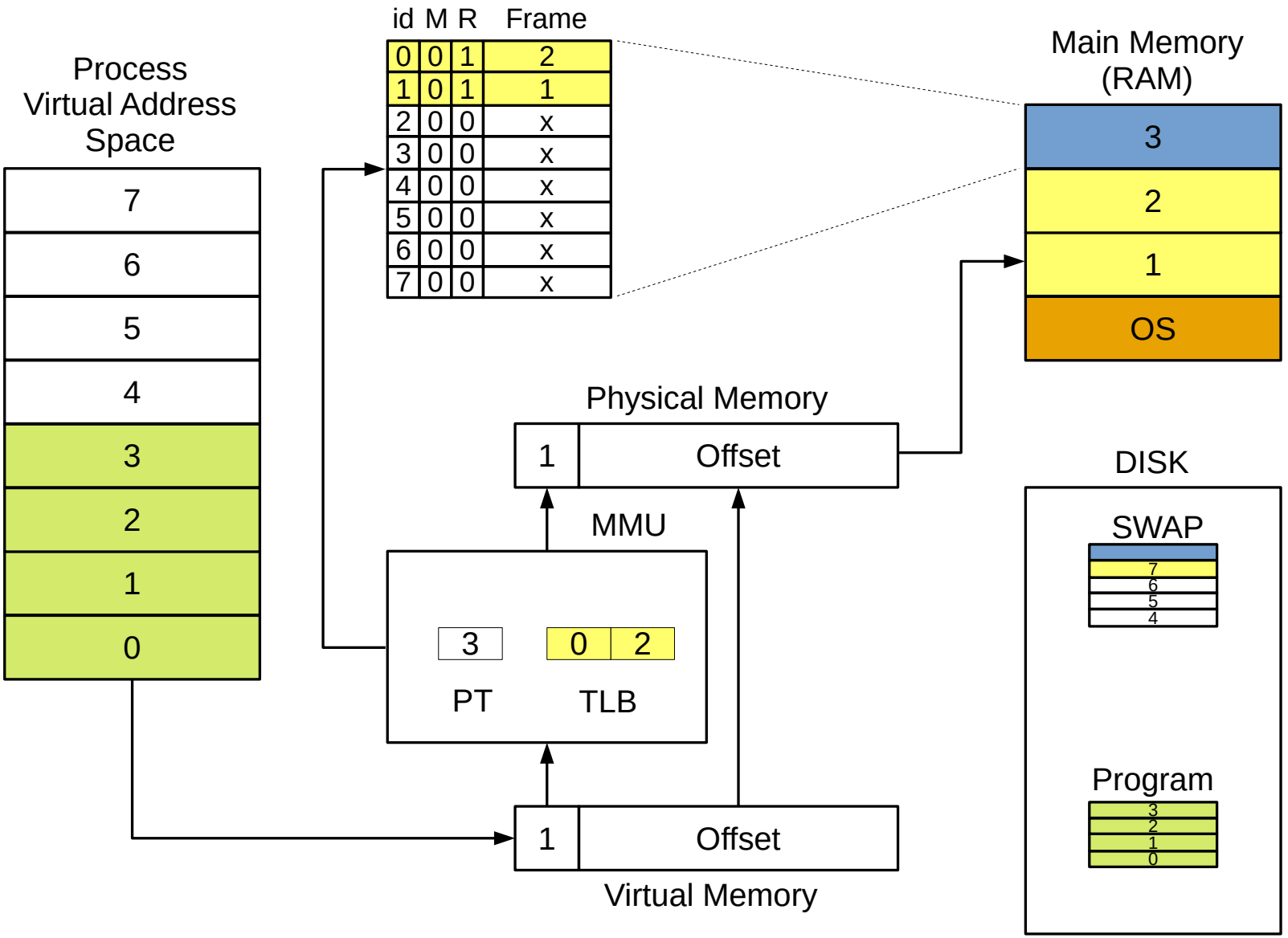

Process
Virtual Address
Space
7
6
5
4
3
2
1
0
id M R Frame
0 0 1 2
1 0 1 1
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 0 0 x
Main Memory
(RAM)
3
2
1
OS
Physical Memory
x Offset
MMU
3
PT
0 2
TLB
0 Offset
Virtual Memory
DISK
SWAP
7
6
5
4
Program
3
2
1
0


Process
Virtual Address
Space
7
6
5
4
3
2
1
0
id M R Frame
0 0 1 2
1 0 1 1
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 0 0 x
Main Memory
(RAM)
3
2
1
OS
Physical Memory
2 Offset
MMU
3
PT
0 2
TLB
DISK
SWAP
7
6
5
4
Program
3
2
1
0
0 Offset
Virtual Memory


Process
Virtual Address
Space
7
6
5
4
3
2
1
0
id M R Frame
0 0 1 2
1 0 1 1
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 0 0 x
Main Memory
(RAM)
3
2
1
OS
Physical Memory
x
Offset
MMU
3
0 2
PT
TLB
1
Offset
Virtual Memory
DISK
SWAP
7
6
5
4
Program
3
2
1
0


Process
Virtual Address
Space
7
6
5
4
3
2
1
0
id M R Frame
0 0 1 2
1 0 1 1
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 0 0 x
Main Memory
(RAM)
3
2
1
OS
Physical Memory
x
Offset
MMU
3
PT
0 2
TLB
TLB – page fault
DISK
SWAP
7
6
5
4
Program
3
2
1
0
1
Offset
Virtual Memory


Process
Virtual Address
Space
7
6
5
4
3
2
1
0
id M R Frame
0 0 1 2
1 0 1 1
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 0 0 x
Main Memory
(RAM)
3
2
1
OS
Physical Memory
1 Offset
MMU
3
PT
0 2
TLB
DISK
SWAP
7
6
5
4
Program
3
2
1
0
1 Offset
Virtual Memory


Process
Virtual Address
Space
7
6
5
4
3
2
1
0
id M R Frame
0 0 1 2
1 0 1 1
2 0 0 x
3 0 0 x
4 0 0 x
5 0 0 x
6 0 0 x
7 0 0 x
Main Memory
(RAM)
3
2
1
OS
Physical Memory
1 Offset
MMU
3
PT
0 2
TLB
1 Offset
Virtual Memory
DISK
SWAP
7
6
5
4
Program
3
2
1
0

Zhrnutie

Zhrnutie

- Virtuálna pamäť = nezávislosť od HW, stabilita a bezpečnosť.
- Segmentovanie umožňuje logicky oddeliť program a dáta do samostatných adresných priestorov.
  - Každý segment adresujeme od 0.
  - každý segment má svoju limitnú adresu čo umožňuje kontrolovať bezpečnosť.
  - Pre programátora je segmentovanie čiastočne transparentné.
- Stránkovanie optimalizuje utilizáciu systému.
  - V RAM sú len tie časti programu a dát, ktoré potrebujeme.
  - Výmena dát medzi diskom a RAM je efektívnejšia.
  - Pre programátora je stránkovanie transparentné.
- HW - MMU a TLB

Čo robiť do ďalšej prednášky

- Znovu prečítať kapitolu 3 z Tanenbauma.