ບົດທີ 8

ການຈັດການໜ່ວຍຄວາມຈຳຈຳລອງ
(Virtual Memory Management)

ເນື້ອໃນຫຍໍ້

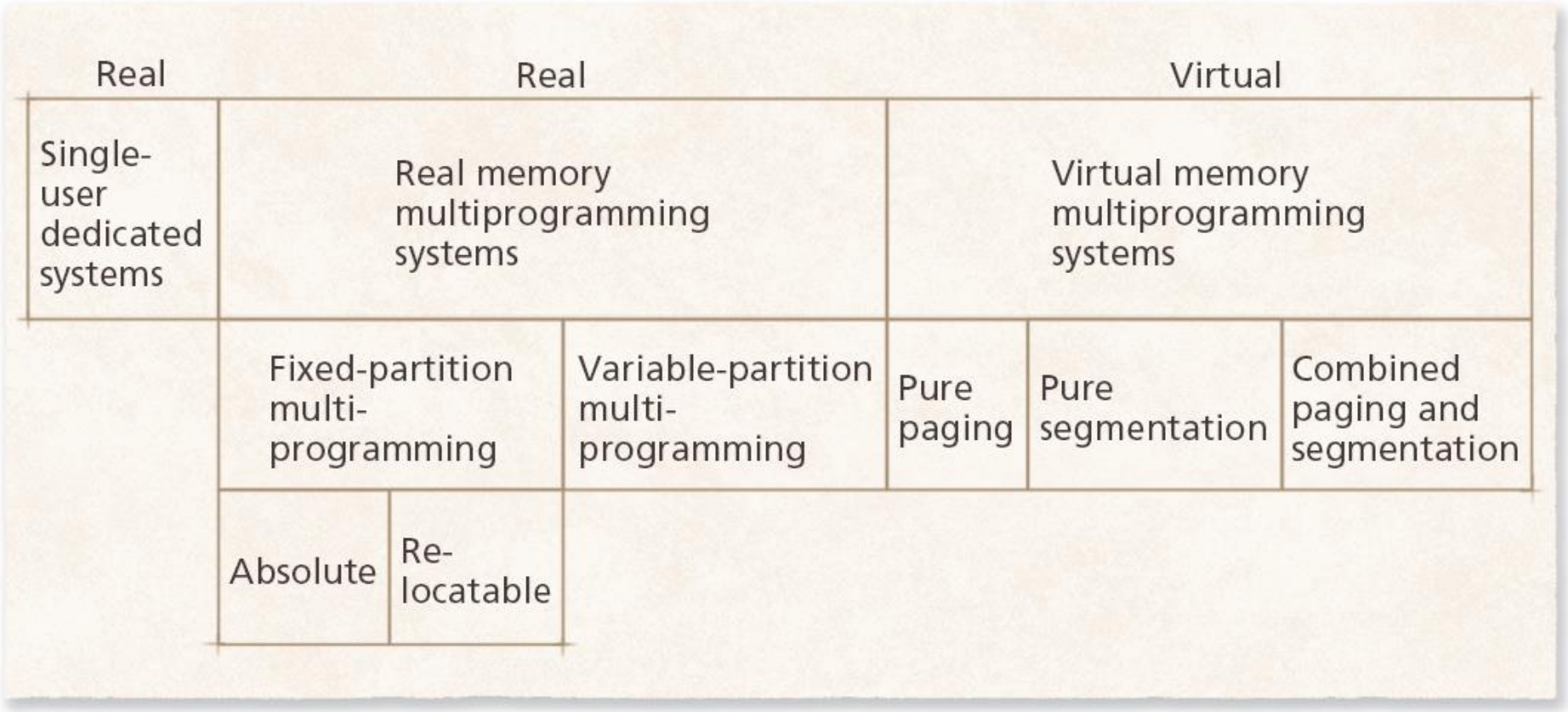
- ສະເໜີເບື້ອງຕົ້ນ
- ແນວຄິດຂອງໜ່ວຍຄວາມຈຳຈຳລອງ
- ການແມັບໜ່ວຍຄວາມຈຳເປັນ Block
- ການແບ່ງໜ້າ
- ການແບ່ງສ່ວນ
- ລະບົບການແບ່ງໜ້າ ແລະ ການແບ່ງສ່ວນ
- ກໍລະນີສຶກສາ: IA-32 Intel Architecture Virtual Memory

ສະເໝີເບື້ອງຕົ້ນ

- ໜ່ວຍຄວາມຈຳຈຳລອງເປັນການສ້າງໜ່ວຍຄວາມຈຳແບບຕັກກະ ຂຶ້ນມາ ເພື່ອເຮັດໃຫ້ມີໜ່ວຍຄວາມຈຳຫຼາຍກວ່າທີ່ມີຢູ່ຈິງ ເຮັດໃຫ້ມີ ເນື້ອທີ່ໜ່ວຍຄວາມຈຳພຽງພໍຕໍ່ການໃຊ້ງານໃນລະບົບ
- ໃນລະບົບໜ່ວຍຄວາມຈຳຈຳລອງຈະມີເລກທີ່ຢູ່ 2 ປະເພດ
  - ເລກທີ່ຢູ່ແບບຈຳລອງ ຫຼື ແບບຕັກກະ (Virtual Address)
    - ເອີ້ນໃຊ້ໂດຍຂະບວນການຕ່າງໆ
  - ເລກທີ່ຢູ່ແບບກາຍຍະພາບ (Physical Address)
    - ເປັນຕົວບອກສະຖານທີ່ເກັບຂໍ້ມູນໃນໜ່ວຍຄວາມຈຳຫຼັກ
- ຜູ້ຈັດການໜ່ວຍຄວາມຈຳເປັນຜູ້ປ່ຽນເລກທີ່ຢູ່ແບບຈຳລອງໄປເປັນ ເລກທີ່ຢູ່ແບບກາຍຍະພາບ

ສະເໜີເບື້ອງຕົ້ນ

- ວິວັດທະນາການໃນການຈັດການໜ່ວຍຄວາມຈຳ



ແນວຄິດຂອງໜ່ວຍຄວາມຈຳຈຳລອງ

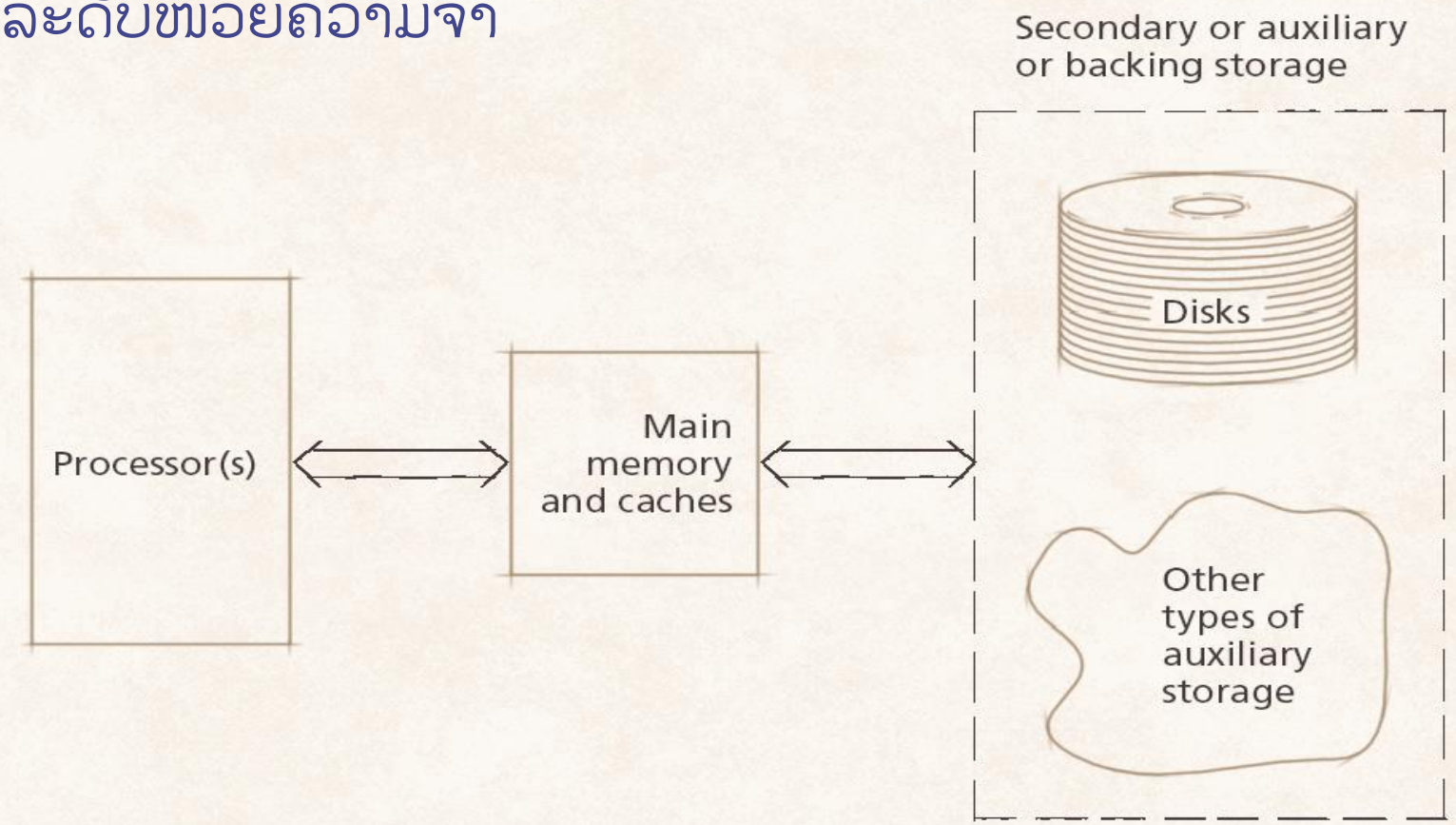
- ຊ່ວງຂອງເລກທີ່ຢູ່ຈຳລອງ (Virtual Address Space), *V*
  - ເປັນຊ່ວງຂອງເລກທີ່ຢູ່ຈຳລອງທີ່ຂະບວນການສາມາດອ້າງອີງໄດ້
- ຊ່ວງຂອງເລກທີ່ຢູ່ກາຍຍະພາບ (Real Address Space), *R*
  - ເປັນຊ່ວງຂອງເລກທີ່ຢູ່ກາຍຍະພາບທີ່ໃຊ້ໄດ້ໃນຄອມພິວເຕີໃດໜຶ່ງ
- ກົນໄກໃນການປ່ຽນເລກທີ່ຢູ່ແບບປ່ຽນແປງໄດ້
  - ເປັນວິທີປ່ຽນຈາກທີ່ຢູ່ແບບຈຳລອງໄປເປັນທີ່ຢູ່ກາຍຍະພາບໃນລະຫວ່າງ ໂປຣແກຣມກຳລັງເຮັດວຽກ

ແນວຄິດຂອງໜ່ວຍຄວາມຈຳຈຳລອງ

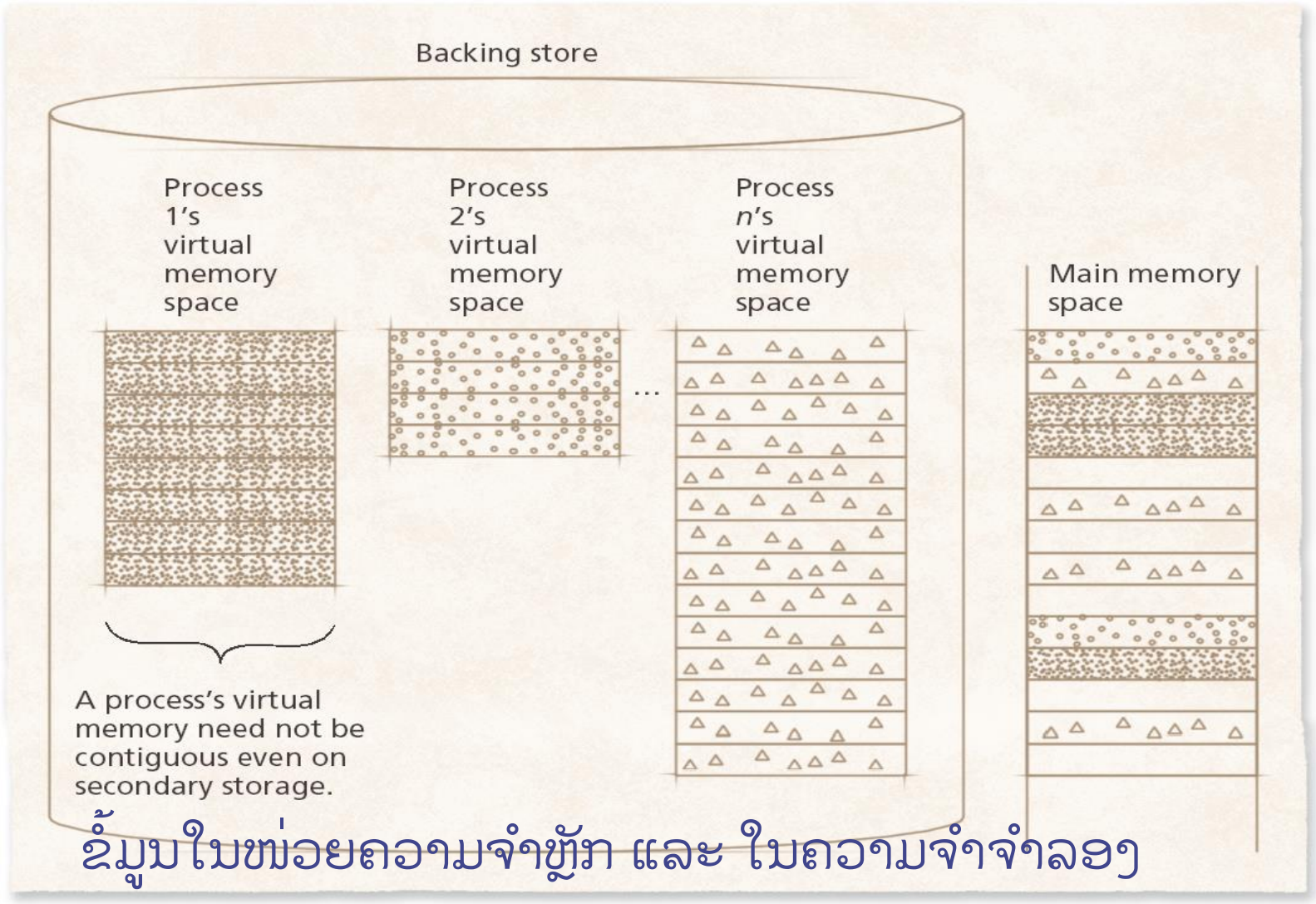
- ເລກທີ່ຢູ່ຈຳລອງຈະມີຈຳນວນຫລາຍກ່ວາເລກທີ່ຢູ່ແບບກາຍຍະພາບ ແລະ ຈະຕ້ອງໄດ້ປ່ຽນເລກທີ່ຢູ່ຈຳລອງໃຫ້ກາຍເປັນເລກທີ່ຢູ່ແບບກາຍຍະພາບເອີ້ນວ່າ Address Mapping
  - ລະບົບປະຕິບັດການຈະເກັບບາງສ່ວນຂອງແຕ່ລະຂະບວນການໄວ້ນອກໜ່ວຍຄວາມຈຳຫຼັກ
  - ລະບົບປະຕິບັດການຈະຍ້າຍຂໍ້ມູນໄປມາລະຫວ່າງໜ່ວຍຄວາມຈຳຫຼັກ ແລະ ໜ່ວຍຄວາມຈຳສຳຮອງ

ແນວຄິດຂອງໜ່ວຍຄວາມຈຳຈຳລອງ

2 ລະດັບໜ່ວຍຄວາມຈຳ



ແນວຄິດຂອງໜ່ວຍຄວາມຈຳຈຳລອງ

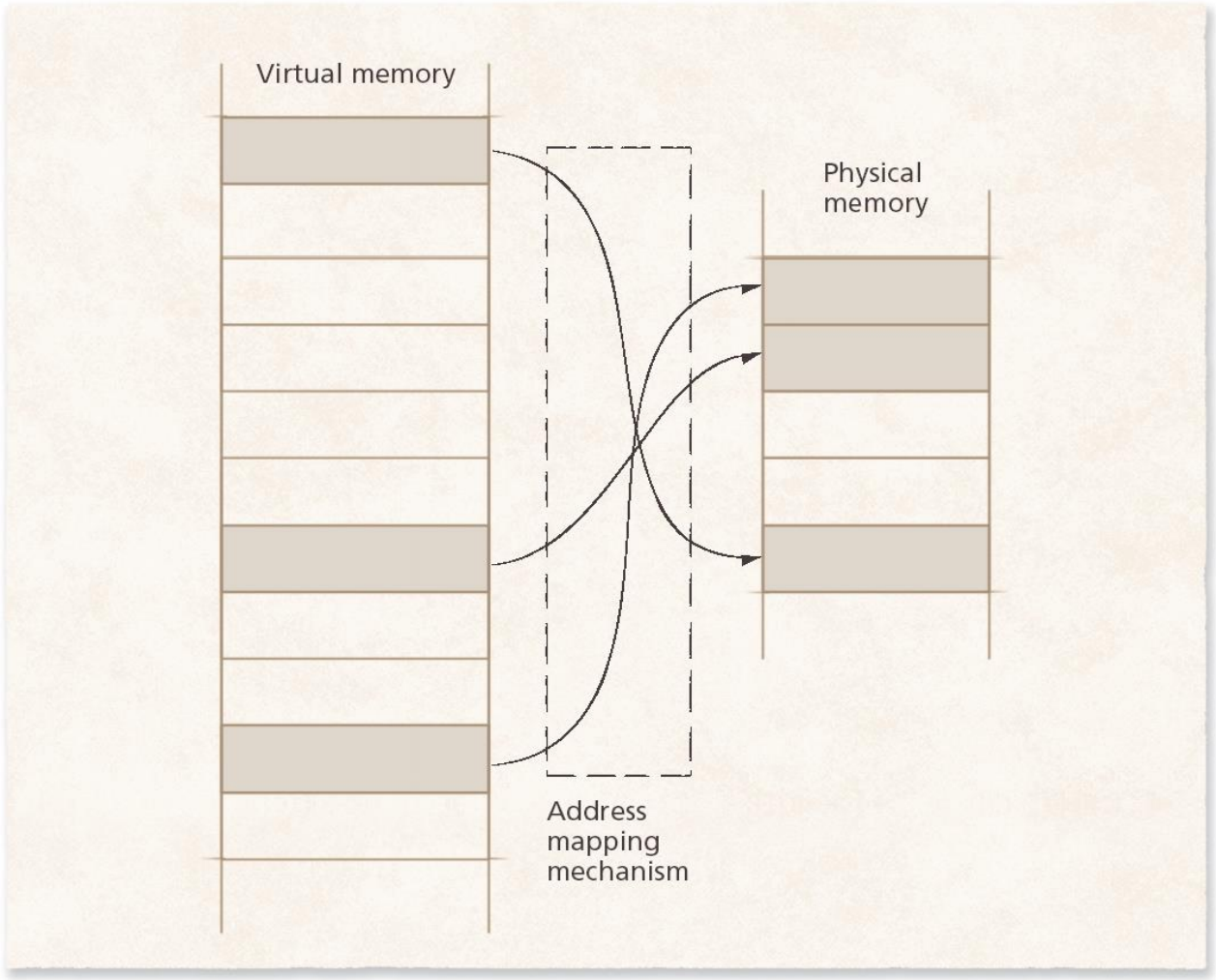


ຂໍ້ມູນໃນໜ່ວຍຄວາມຈຳຫຼັກ ແລະ ໃນຄວາມຈຳຈຳລອງ

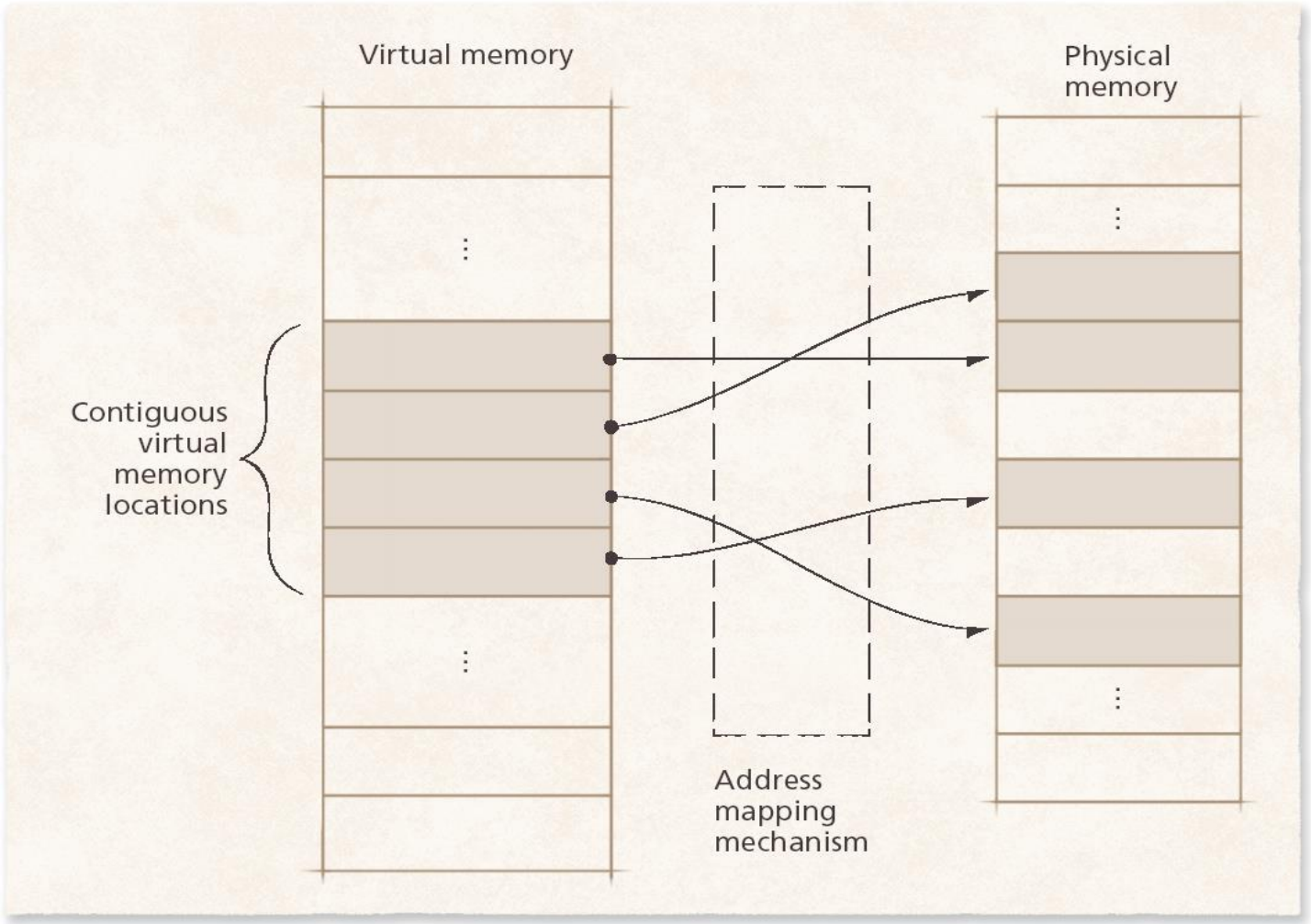
ການແບ່ງໜ່ວຍຄວາມຈຳເປັນຫຼາຍສ່ວນ

- ເປັນການຈັດກຸ່ມຂໍ້ມູນເປັນຫຼາຍສ່ວນເອີ້ນວ່າ Block
- ກຳໜົດວ່າເຂດໃດໃນໜ່ວຍຄວາມຈຳຈຳລອງທີ່ກຳລັງຖືກເກັບໄວ້ໃນໜ່ວຍຄວາມຈຳຫຼັກ ແລະ ເກັບໄວ້ບ່ອນໃດ
- ການລຽງລຳດັບຕໍ່ເນື່ອງກັນຂອງຂໍ້ມູນຢູ່ໃນໜ່ວຍຄວາມຈຳຈຳລອງແມ່ນໃດ ອາດຈະບໍ່ໄດ້ລຽງລຳດັບຕໍ່ເນື່ອງກັນຄືແມ່ນນັ້ນຢູ່ໃນໜ່ວຍຄວາມຈຳແບບກາຍຍະພາບ

ການແບ່ງໜ່ວຍຄວາມຈຳເປັນຫຼາຍສ່ວນ



ການແບ່ງໜ່ວຍຄວາມຈຳເປັນຫຼາຍສ່ວນ



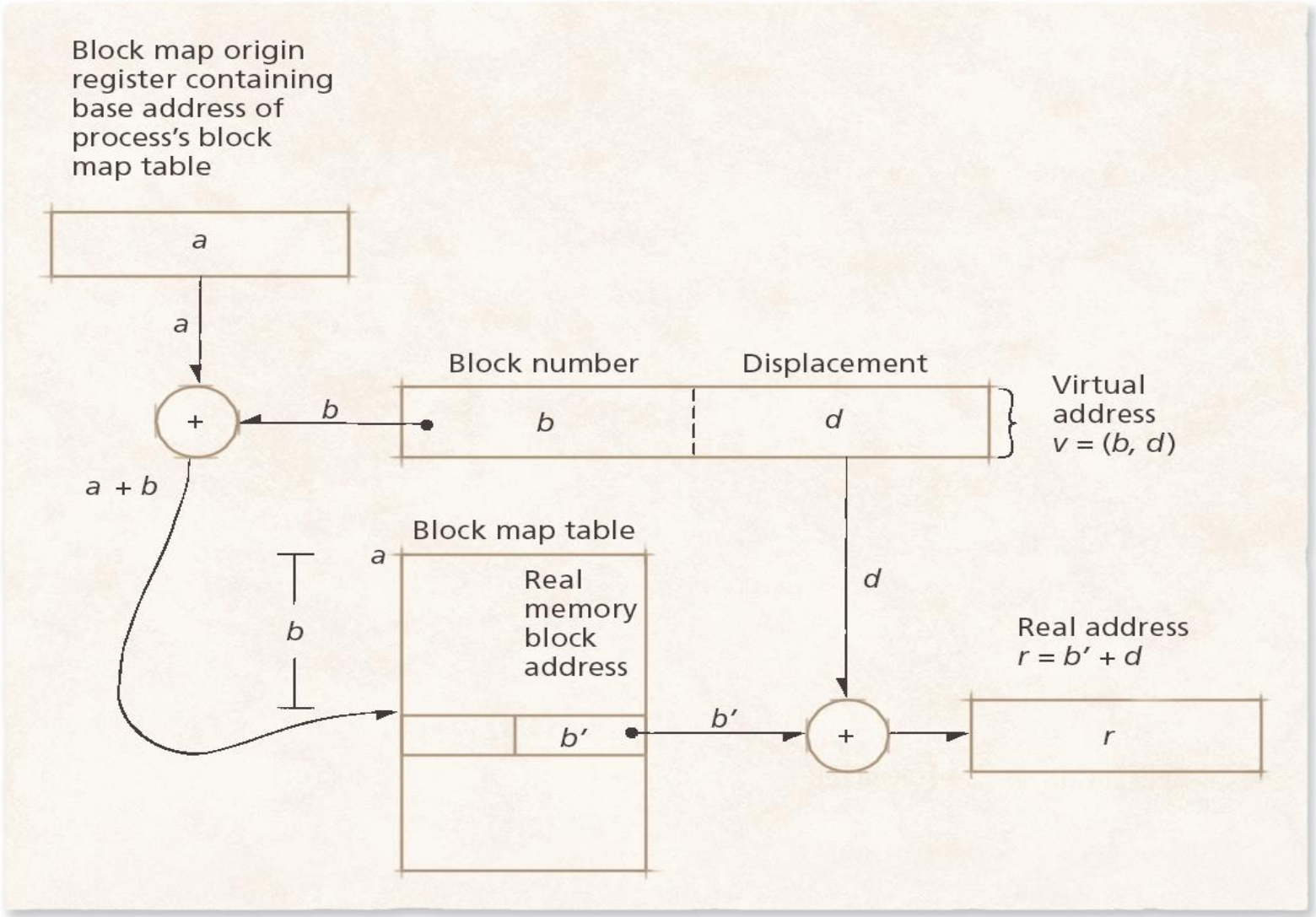
ການແບ່ງໜ່ວຍຄວາມຈຳເປັນຫຼາຍສ່ວນ

- ການແບ່ງໜ່ວຍຄວາມຈະອອກເປັນສ່ວນມີຫຼາຍກໍລະນີດັ່ງນີ້
  - ການແບ່ງໜ່ວຍຄວາມຈຳອອກເປັນໜ້າ (Pages)
    - ຂະໜາດຂອງແຕ່ລະໜ້າຈະບໍ່ປ່ຽນແປງ ແລະ ມີຂະໜາດເທົ່າກັນ
    - ເທັກນິກທີ່ໃຊ້ເອີ້ນວ່າ paging
  - ການແບ່ງໜ່ວຍຄວາມຈຳອອກເປັນສ່ວນ (Segments)
    - ຂະໜາດຂອງແຕ່ລະສ່ວນອາດຈະບໍ່ເທົ່າກັນ
    - ເທັກນິກທີ່ໃຊ້ເອີ້ນວ່າ segmentation
- ການແບ່ງໜ່ວຍຄວາມຈຳເປັນຫຼາຍສ່ວນ (Block mapping)
  - ລະບົບຈະສະແດງທີ່ຢູ່ຂອງແຕ່ລະຂໍ້ມູນເປັນຄູ່ຂອງໝາຍເລກສ່ວນ ແລະ ລຳດັບຂອງແຕ່ລະຂໍ້ມມູທີ່ຢູ່ໃນແຕ່ລະກ້ອນ ເພື່ອເປັນການອ້າງອີງຫາຂໍ້ມູນໃດໜຶ່ງ

ການແບ່ງໜ່ວຍຄວາມຈຳເປັນຫຼາຍສ່ວນ

- ຕົວຢ່າງໃຫ້ທີ່ຢູ່ຂອງຄວາມຈຳຈຳລອງ v = (b, d) ໃດໜຶ່ງ
  - Block map origin register ຈະເກັບທີ່ຢູ່ທຳອິດ a ຂອງຕາຕະລາງ block
  - ໝາຍເລກ block, b, ໄດ້ຖືກບວກເຂົ້າກັບ a ເພື່ອກຳໜົດໝາຍເລກ block ຈາກຕາຕະລາງ block
  - ຈະໄດ້ທີ່ຢູ່ເລີ່ມຕົ້ນຂອງ block b, b́, ຂອງໜ່ວຍຄວາມຈຳຫຼັກ
  - ເອົາຄ່າ d ບວກໃສ່ b́ ຈະໄດ້ທີ່ຢູ່ກາຍຍະພາບ, r ອອກມາ

ການແບ່ງໜ່ວຍຄວາມຈຳເປັນຫຼາຍສ່ວນ



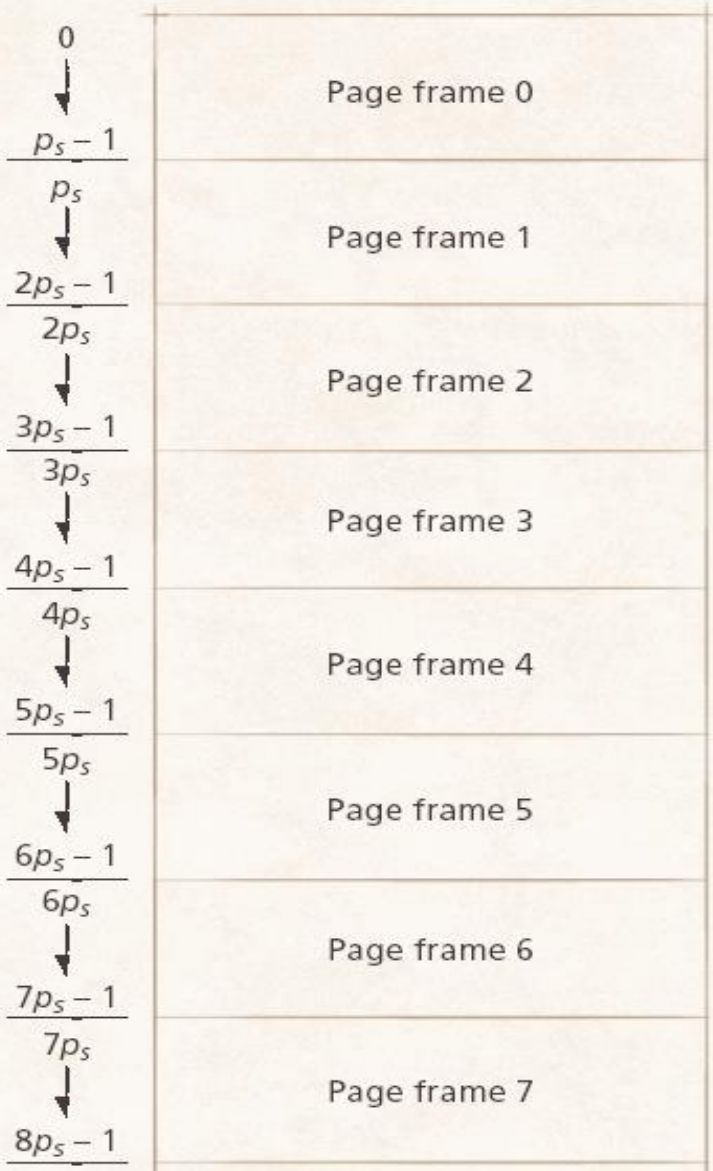
ການແບ່ງໜ້າ

- ເປັນການແບ່ງເນື້ອທີ່ໜ່ວຍຄວາມຈຳ ແລະ ຂະບວນການອອກເປັນໜ້າ (Paging) ແຕ່ລະໜ້າຈະມີຂະໜາດເທົ່າກັນ
- ເປັນການແບ່ງສ່ວນທີ່ມີຂະໜາດຄົງທີ
- ເລກທີ່ຢູ່ຈຳລອງໃນລະບົບນີ້ຈະເປັນລຳດັບຂອງຄູ່ (p, d)
  - p ເປັນໝາຍເລກໜ້າທີ່ບັນຈຸຂໍ້ມູນຢູ່
  - d ເປັນໝາຍເລກຂອງແຕ່ລະຂໍ້ມູນຢູ່ໃນແຕ່ລະໜ້າ

ການແບ່ງໜ້າ

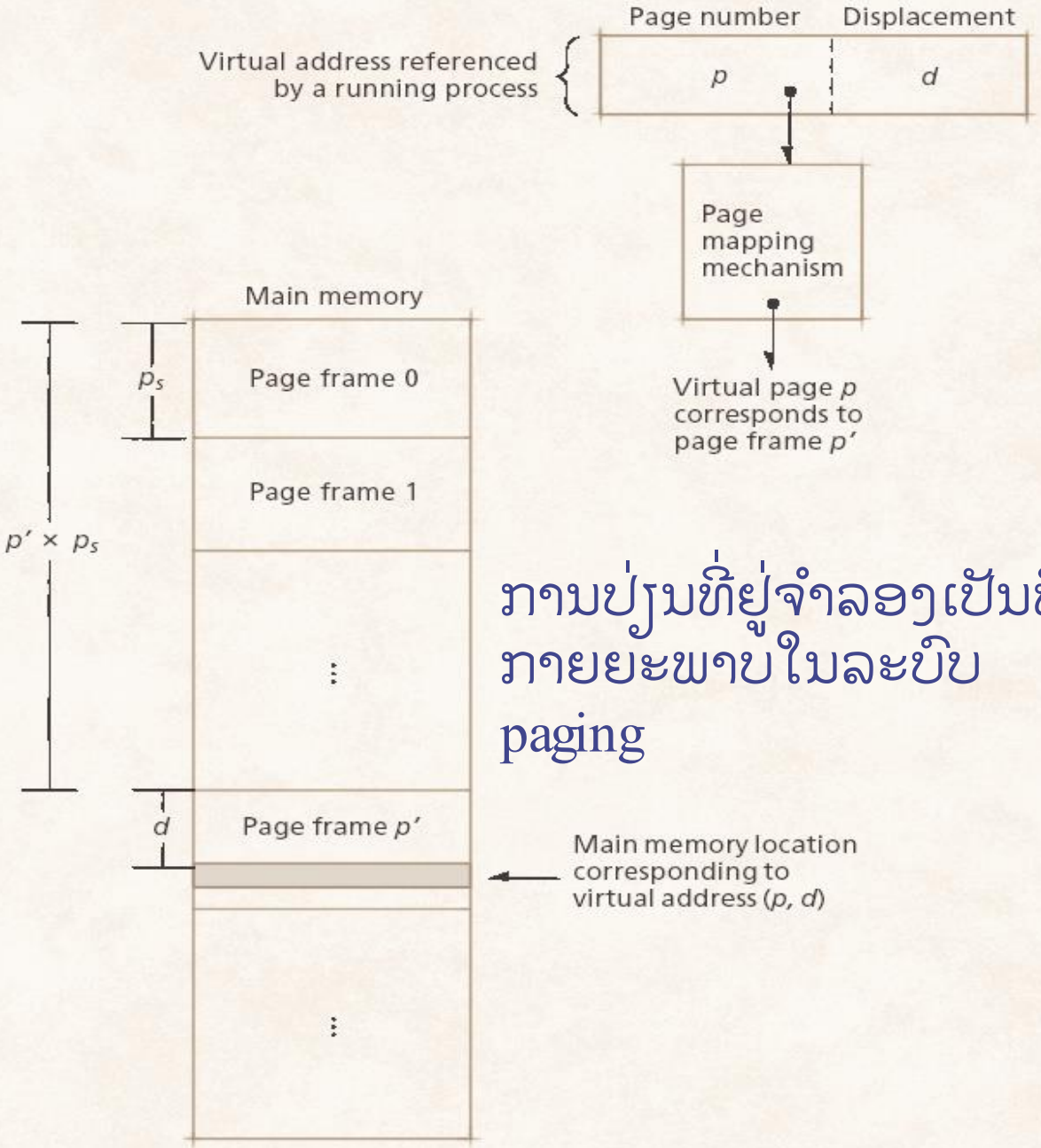
- ວິທີການພື້ນຖານ
  - ໜ່ວຍຄວາມຈຳແບບກາຍຍະພາບຈະຖືກແບ່ງອອກເປັນຫຼາຍສ່ວນທີ່ມີຂະໜາດເທົ່າກັນເອີ້ນວ່າ Frame
  - ໜ່ວຍຄວາມຈຳແບບຕັກກະກ່ຳຖືກແບ່ງອອກເປັນຫຼາຍສ່ວນທີ່ເອີ້ນວ່າ Page ແລະ ມີຂະໜາດເທົ່າກັບ Frame
  - ຂະໜາດຂອງໜ້າທີ່ນິຍົມໃຊ້ແມ່ນ 2 – 8 KB
    - ບ່າງໜ່ວຍປະມວນຜົນ ແລະ kernel ສະໜັບສະໜູນການມີຂະໜາດໜ້າຫຼາຍໆຂະໜາດເຊັ່ນ: Solaris ມີຂະໜາດ 4 ແລະ 8 KB ຂຶ້ນກັບປະເພດຂໍ້ມູນ

ภาม



Page frame number	Page frame size	Range of physical memory addresses
0	p_s	$0 \rightarrow p_s - 1$
1	p_s	$p_s \rightarrow 2p_s - 1$
2	p_s	$2p_s \rightarrow 3p_s - 1$
3	p_s	$3p_s \rightarrow 4p_s - 1$
4	p_s	$4p_s \rightarrow 5p_s - 1$
5	p_s	$5p_s \rightarrow 6p_s - 1$
6	p_s	$6p_s \rightarrow 7p_s - 1$
7	p_s	$7p_s \rightarrow 8p_s - 1$
⋮		

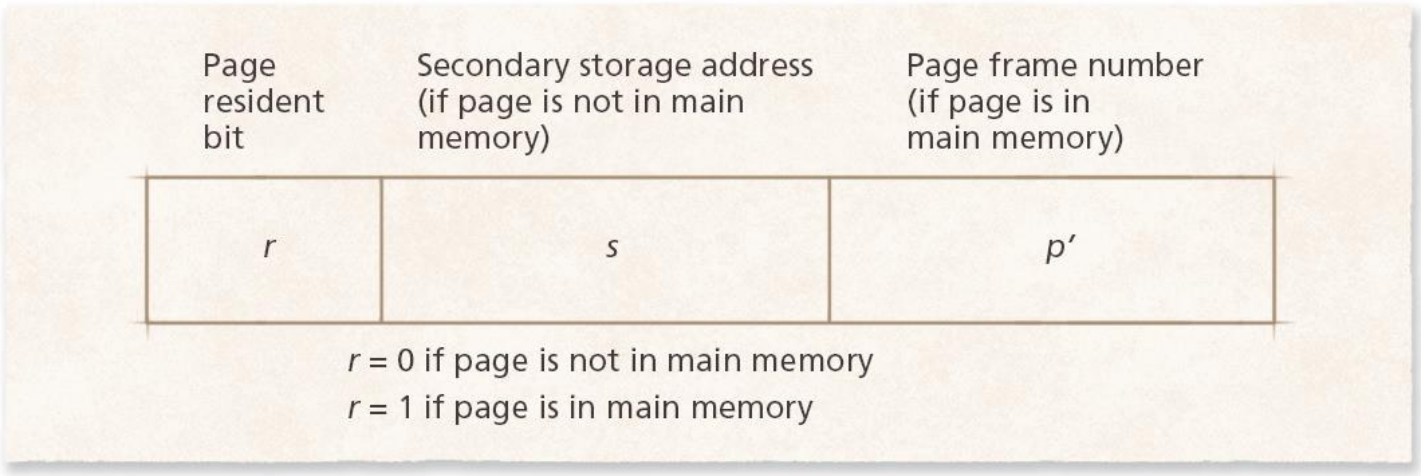
ໜ່ວຍຄວາມຈຳຫຼັກໄດ້ຖືກແບ່ງອອກເປັນຫຼາຍສ່ວນ



ການປ່ຽນທີ່ຢູ່ຈຳລອງເປັນທີ່ຢູ່ກາຍຍະພາບໃນລະບົບ paging

ການແບ່ງໜ້າ

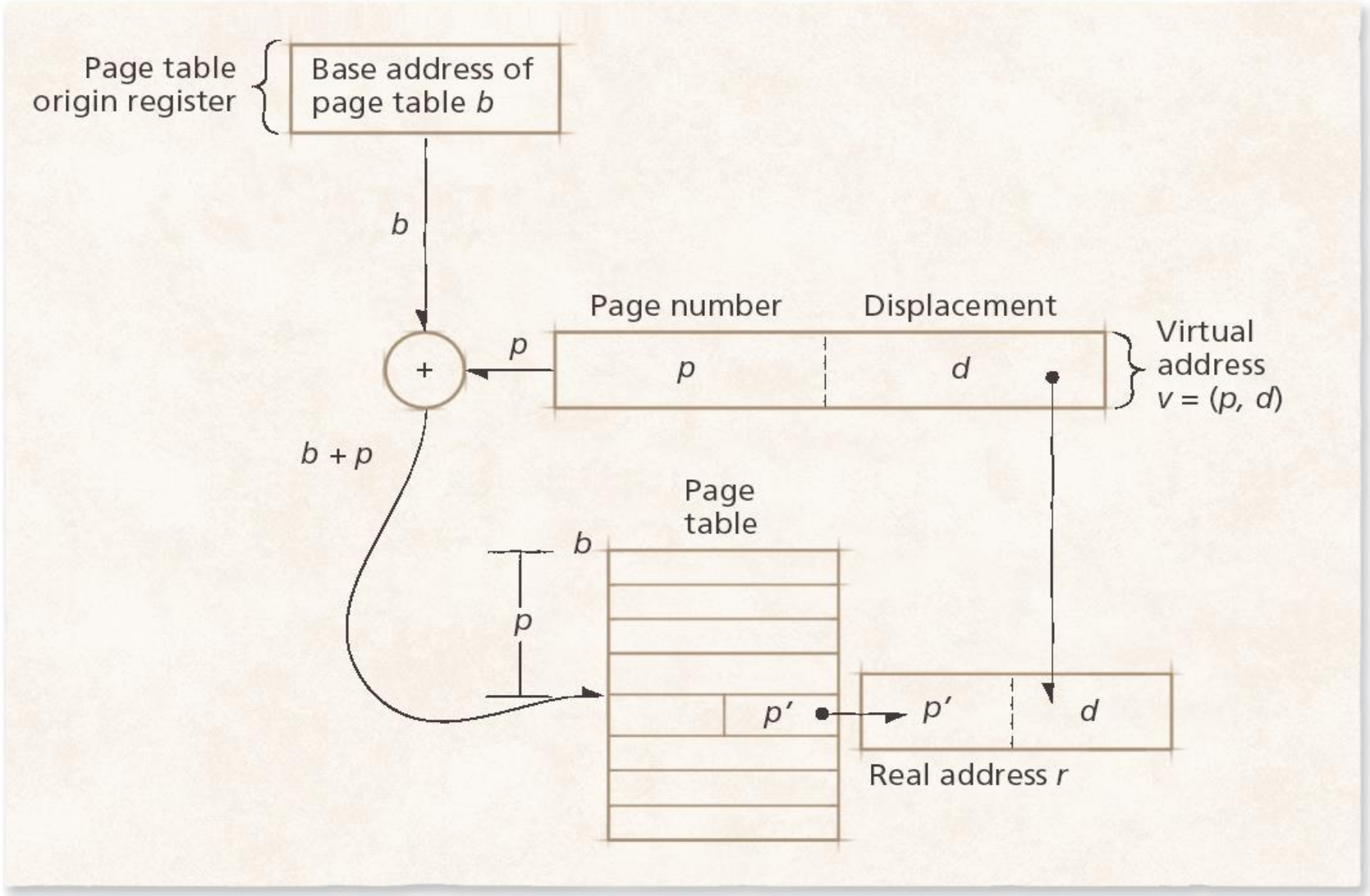
- ຕາຕະລາງເລກໜ້າ (Page table entry (PTE))
  - ເປັນຕົວກຳໜົດວ່າ ໜ້າທີ p ໃນໜ່ວຍຄວາມຈຳຈຳລອງ ກົງກັບໜ້າ ṕ ໃນໜ່ວຍຄວາມຈຳຫຼັກ
  - ບັນຈຸບິດສັງກັດຢູ່ (resident bit) ເພື່ອສະແດງວ່າໜ້າດັ່ງກ່າວເກັບຢູ່ໃນໜ່ວຍຄວາມຈຳຫຼັກ
  - ຖ້າໜ້າດັ່ງກ່າວເກັບຢູ່ໃນໜ່ວຍຄວາມຈຳຫຼັກ, PTE ຈະເກັບໝາຍເລກຂອງ page's frame
  -



ການແບ່ງໜ້າ

- ການປ່ຽນເລກທີ່ຢູ່ຈຳລອງເປັນເລກທີ່ຢູ່ກາຍະພາບແບບໂດຍກົງ
  - ຄືກັບ Block mapping
  - ເມື່ອຂະບວນການອ້າງອີງທີ່ຢູ່ຈຳລອງ v = (p, d)
    - DAT ຈະບອກທີ່ຢູ່ເລີ່ມຕົ້ນຂອງຕາຕະລາງເລກໜ້າ , b, ເພື່ອອ້າງອີງໄປຫາໜ້າ, p
    - b + p ແມ່ນຕຳແໜ່ງຂອງໜ້າ p ທີ່ໃນ PTE
    - ລະບົບຈະເຊື່ອມຕໍ່ ṕ ກັບ, d, ໃຫ້ເປັນທີ່ຢູ່ແບບກາຍຍະພາບ, r

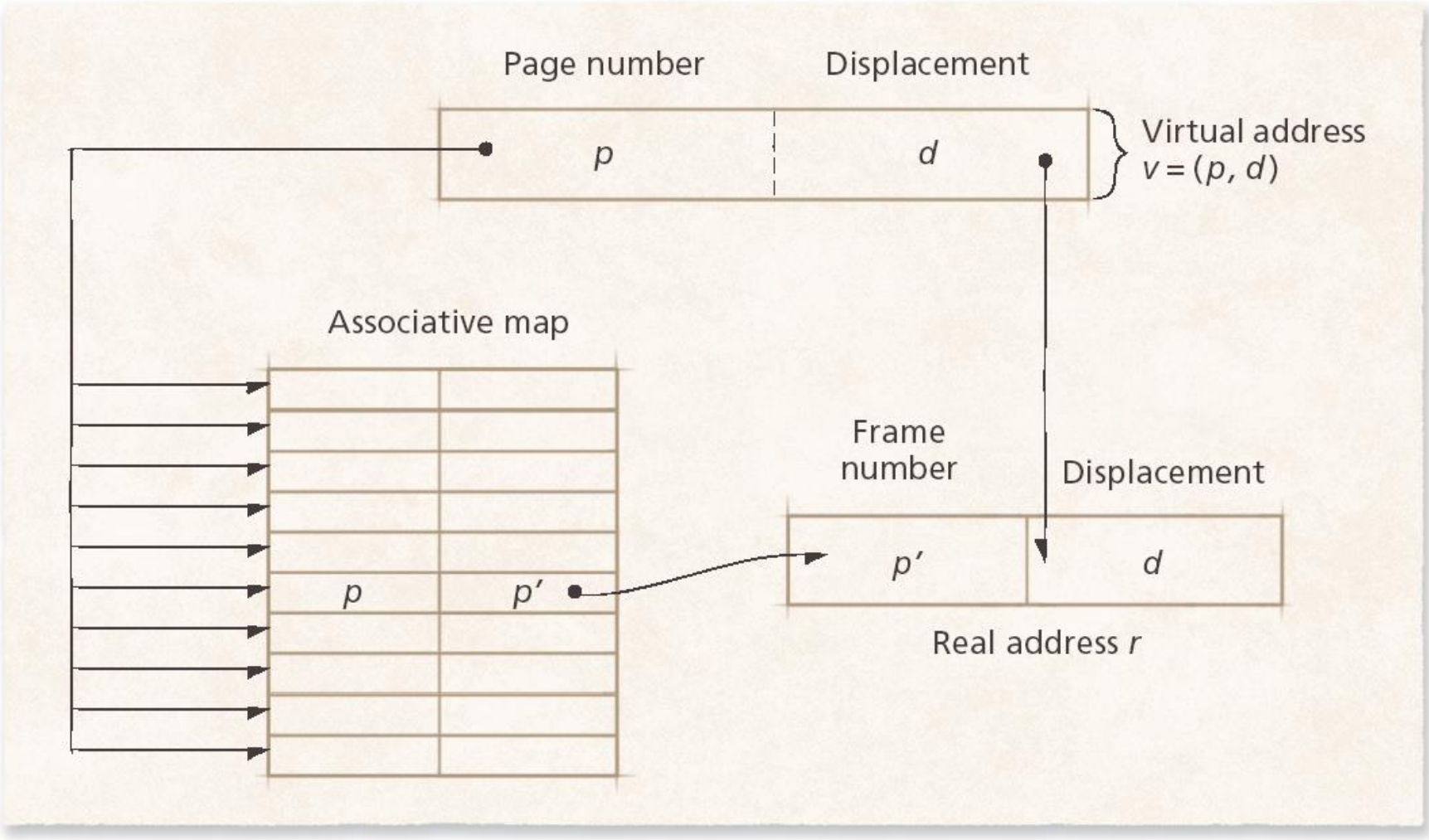
ການແບ່ງໜ້າ


Page table origin register
Base address of page table b
b
Page number
Displacement
p
d
Virtual address v = (p, d)
+
p
b + p
Page table
b
p
p′
p′
d
Real address r

ການແບ່ງໜ້າ

- ການປ່ຽນເລກທີ່ຢູ່ຈຳລອງເປັນເລກທີ່ຢູ່ກາຍະພາບແບບເຮັດເທື່ອດຽວ
  - ເປັນການເອົາທັງໝົດຕາຕະລາງເລກໜ້າໄປເກັບໄວ້ໃນໜ່ວຍຄວາມຈຳຄວາມໄວສຸງ(cache)
  - ເຮັດໃຫ້ຄວາມໄວໃນການປ່ຽນເພີ່ມຂຶ້ນ (ມີປະສິດທິພາບສຸງ)
  - ທຸກໆແຖວໃນໜ່ວຍຄວາມຈຳສາມາດຄົ້ນຫາພ້ອມກັນໄດ້
  - ສິ້ນເປືອງຫຼາຍ ລາຄາແພງ

ການແບ່ງໜ້າ

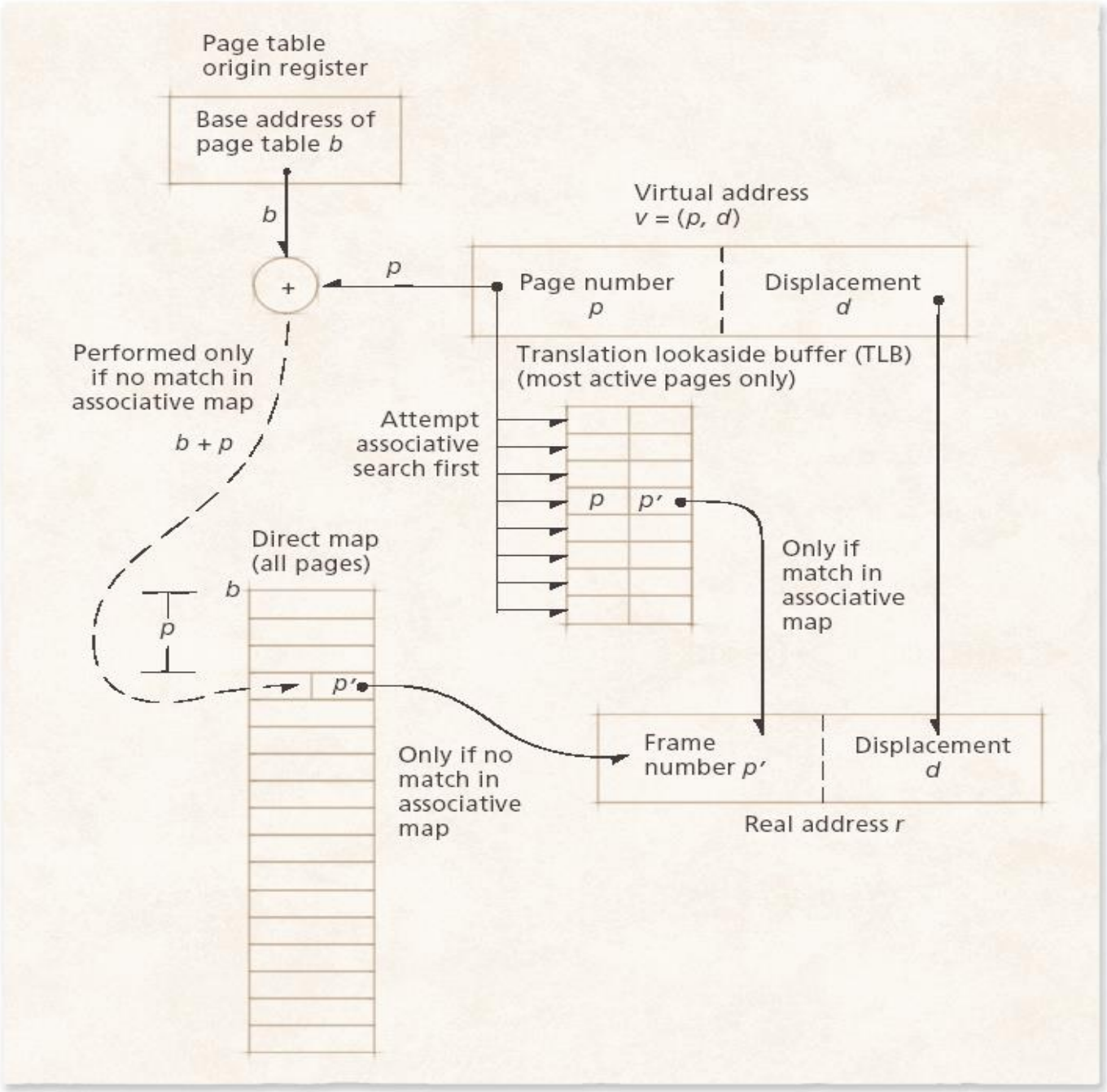

Page number
Displacement
p
d
Virtual address
v = (p, d)
Associative map
p
p′
Frame
number
Displacement
p′
d
Real address r

ການແບ່ງໜ້າ

- ການປ່ຽນເລກທີ່ຢູ່ຈຳລອງເປັນເລກທີ່ຢູ່ກາຍະພາບແບບປະສົມ
  - ເຮັດໃຫ້ສົມດູນລະຫວ່າງປະສິດທິພາບ ແລະ ມູນຄ່າໃນການໃຊ້ຈ່າຍ
  - ຂໍ້ມູນທັງໝົດກ່ຽວກັບການປ່ຽນແມ່ນເກັບໄວ້ໃນຕາຕະລາງເລກໜ້າ
  - ຂໍ້ມູນທີ່ຫາກໍ່ຖືກປ່ຽນຫວ່າງມໍ່ໆນີ້ຈະເກັບໄວ້ໃນ TLB
  - ຖ້າຄົ້ນຫາໃນ TLB ບໍ່ເຫັນຈະມາຄົ້ນຫາໃນຕາຕະລາງເລກໜ້າ
  - ເຮັດໃຫ້ປະສິດທິພາບດີ ໃນຂະນະທີ່ການສົ້ນເປືອງຂະໜາດຂອງຄວາມຈຳສແບບພິເສດໜ້ອຍລົງ

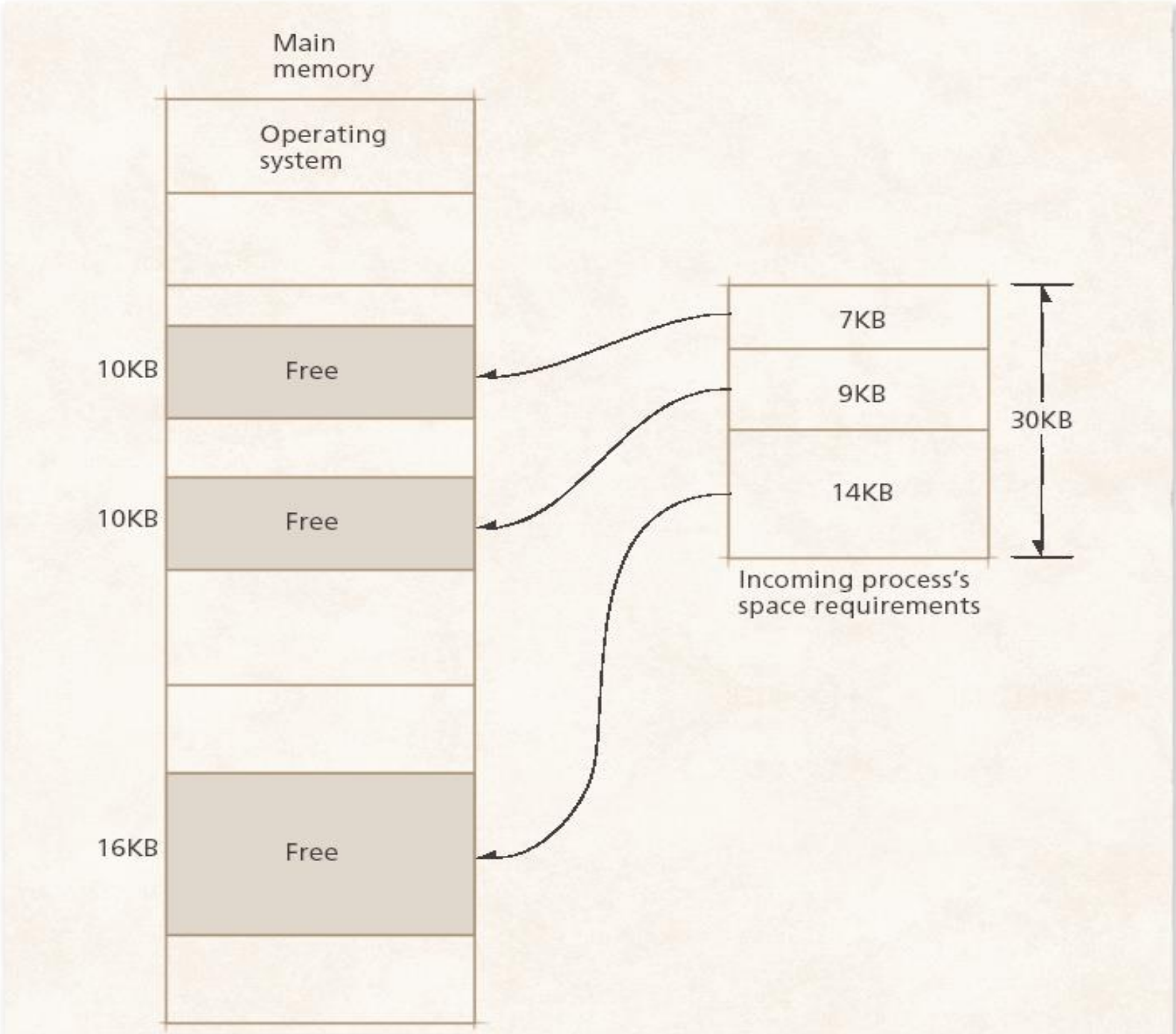
ການແ

- ການ


Page table
origin register
Base address of
page table b
b
+
p
Virtual address
v = (p, d)
Page number
p
Displacement
d
Translation lookaside buffer (TLB)
(most active pages only)
Performed only
if no match in
associative map
b + p
Attempt
associative
search first
p
p′
Only if
match in
associative
map
Direct map
(all pages)
b
p
p′
Only if no
match in
associative
map
Frame
number p′
Displacement
d
Real address r

ການແບ່ງສ່ວນ

- Segment
  - ຂໍ້ມູນ ແລະ ຄຳສັ່ງຂອງໂປຣແກຣມໃດໜຶ່ງໄດ້ຖືກແບ່ງອອກເປັນ block ທີ່ເອີ້ນວ່າ segments.
  - ແຕ່ລະ segment ຈະປະກອບດ້ວຍທີ່ຢູ່ທີ່ຕໍ່ເນື່ອງກັນໄປ.
  - ແຕ່ລະ segment ບໍ່ຈຳເປັນຕ້ອງມີຂະໜາດເທົ່າກັນ ແລະ ກໍ່ບໍ່ຈຳເປັນຕ້ອງເກັບລຽນຕໍ່ເນື່ອງຕິດກັນໄປໃນໜ່ວຍຄວາມຈຳຫຼັກ.
  - ຜົນດີຂອງ Segmentation ທີ່ເໜືອກວ່າ paging ແມ່ນກຳນົດຂະໜາດຕາມຄວາມເໝາະສົມໄດ້
  - ມີແຕ່ Segments ທີ່ໂປຣແກຣມໃດໜຶ່ງຕ້ອງການເພື່ອເຮັດວຽກໃນເວລາໃດໜຶ່ງເທົ່ານັ້ນທີ່ຖືກເກັບໄວ້ໃນໜ່ວຍຄວາມຈຳ; ສ່ວນ Segments ອື່ນໆທີ່ເຫຼືອ ຈະຖືກເກັບໄວ້ໃນໜ່ວຍຄວາມຈຳສຳຮອງ


Main memory
Operating system
10KB
Free
10KB
Free
16KB
Free
7KB
9KB
14KB
30KB
Incoming process's space requirements

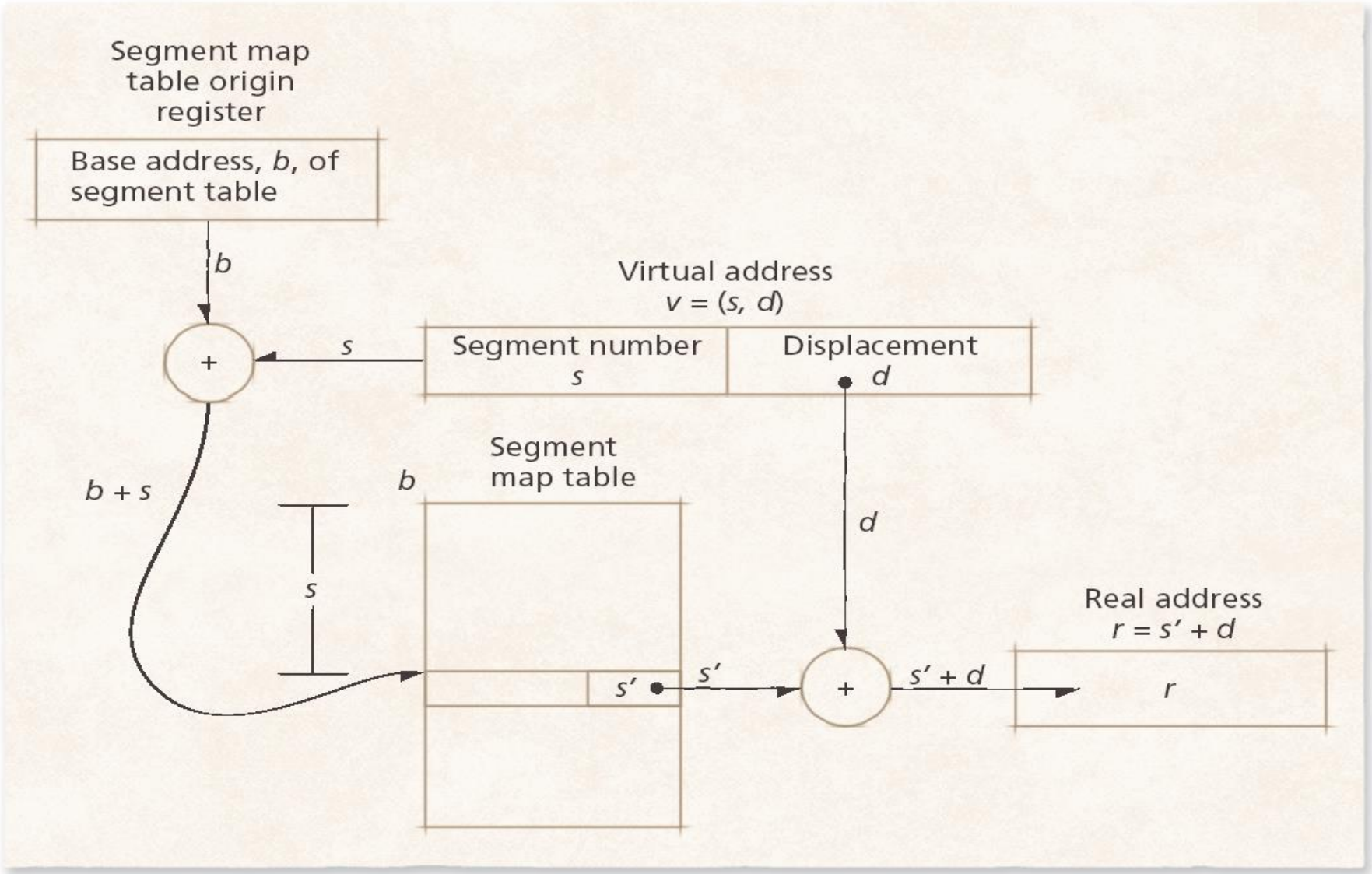
ການແບ່ງສ່ວນ

- ທີ່ຢູ່ຂອງໜ່ວຍຄວາມຈຳທຽມແບບ Segmentation
  - ເປັນຄູ່ລຳດັບຕົວປະສານ v = (s, d)
  - s ແມ່ນໝາຍເລກ Segment ໃນໜ່ວຍຄວາມຈຳທຽມ ທີ່ມີຂໍ້ມູນທີ່ຖືກອ້າງອີງຢູ່
  - d ແມ່ນເລກລຳດັບຂໍ້ມູນທີ່ຢູ່ພາຍໃນ Segment ໃດໜຶ່ງ

ການແບ່ງສ່ວນ

- ການປ່ຽນທີ່ຢູ່ໃນລະບົບ Segmentation ແບບໂດຍກົງ
  - ຂະບວນການໃດໜຶ່ງຈະອ້າງອີງຫາທີ່ຢູ່ທຽມ v = (s, d) ເພື່ອຊອກຫາວ່າ Segment ດັ່ງກ່າວຢູ່ບ່ອນໃດໃນໜ່ວຍຄວາມຈຳຫຼັກ.
  - ລະບົບບອກໝາຍເລກ Segment, s, ໃສ່ທີ່ຢູ່ທຳອິດຂອງຕາຕະລາງການປ່ຽນ Segment, b, ທີ່ເກັບຢູ່ໃນ segment map table oringin register
  - ລະບົບຈະບອກລຳດັບຂອງຂໍ້ມູນ, d, ໃສ່ທີ່ຢູ່ເລີ່ມຕົ້ນດັ່ງກ່າວ ເພື່ອສ້າງເປັນທີ່ຢູ່ຈິງໃນໜ່ວຍຄວາມຈຳຫຼັກ.

ການແບ່ງສ່ວນ

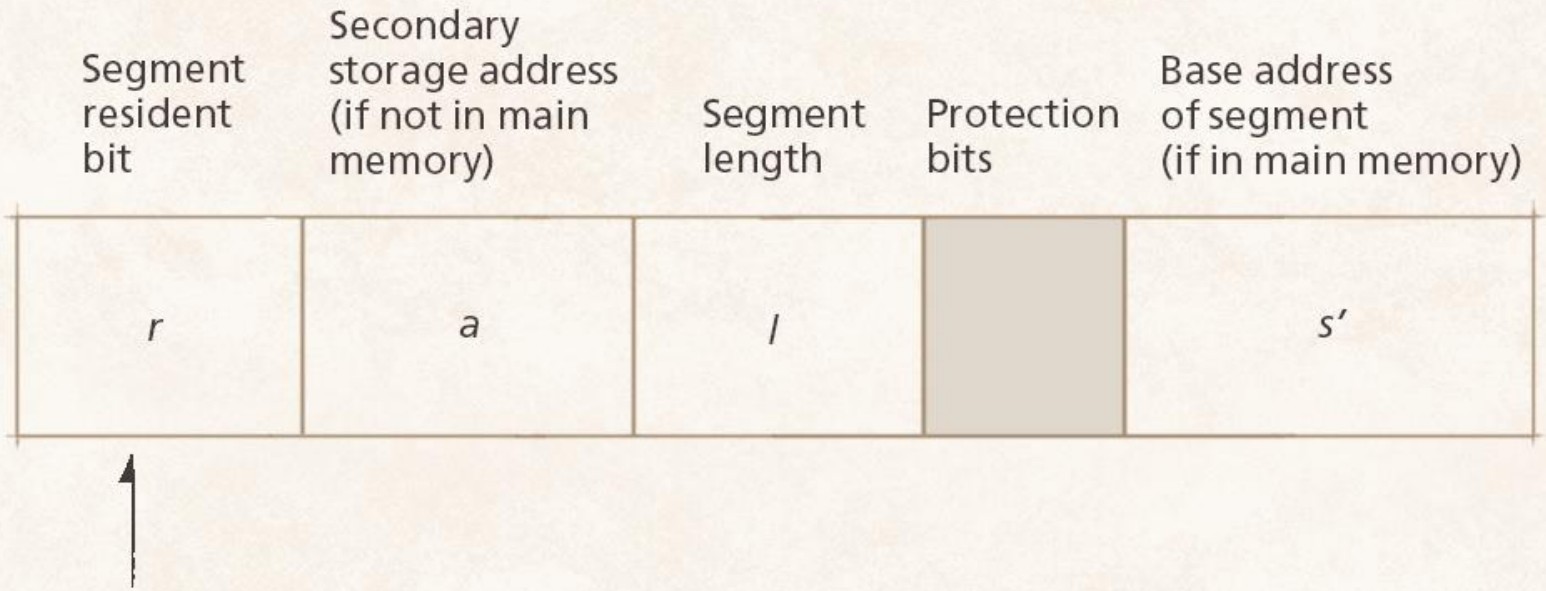

Segment map table origin register
Base address, b, of segment table
b
Virtual address
v = (s, d)
+
s
Segment number
s
Displacement
d
b + s
b
Segment map table
d
s
Real address
r = s′ + d
s′
s′
+
s′ + d
r

ການແບ່ງສ່ວນ

- ຕາຕະລາງການປ່ຽນ Segment
  - ສະແດງໃຫ້ເຫັນວ່າ Segment S ເລີ່ມຕົ້ນຢູ່ທີ່ທີ່ຢູ່ຈິງ S' ໃດໜຶ່ງ
  - ບັນຈຸບິດບອກທີ່ຢູ່ໃດໜຶ່ງວ່າຢູ່ໃນໜ່ວຍຄວາມຈຳຫຼັກຫຼືບໍ່
    - ຖ້າຢູ່ໃນໜ່ວຍຄວາມຈຳຫຼັກ ມັນຈະເກັບທີ່ຢູ່ເລີ່ມຕົ້ນຂອງ Segment
    - ຖ້າບໍ່ຢູ່ ມັນຈະເກັບທີ່ຢູ່ຂອງ Segment ນັ້ນໃນໜ່ວຍຄວາມຈຳສຳຮອງ
  - ນອກຈາກນັ້ນ ຍັງບັນຈຸບ່ອນເກັບຄວາມຍາວທີ່ສະແດງໃຫ້ເຫັນຂະໜາດຂອງ Segment
    - ສາມາດໃຊ້ເພື່ອປ້ອງກັນບໍ່ໃຫ້ຂະບວນການອ້າງອີງທີ່ຢູ່ເກີນຂອບເຂດ

ການແບ່ງສ່ວນ

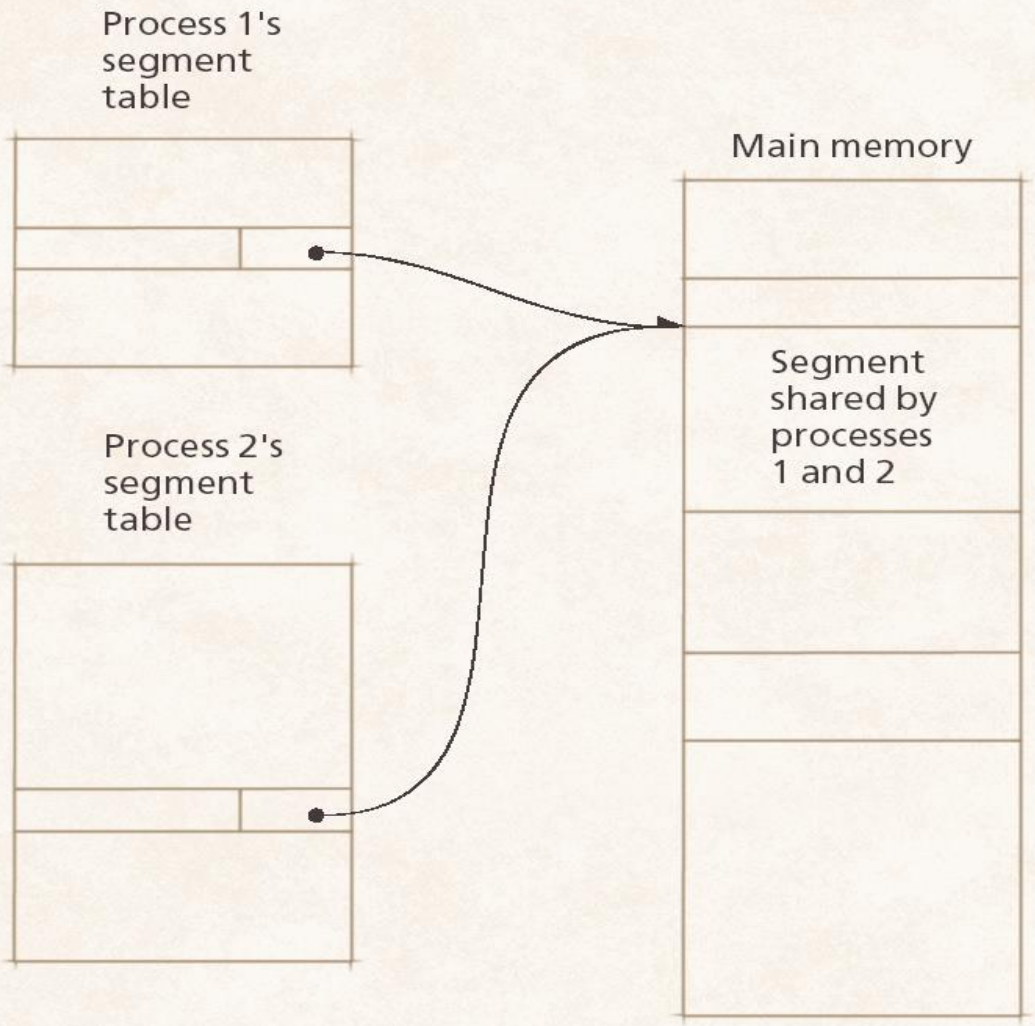
- ຕາຕະລາງການປ່ຽນ Segment



ການແບ່ງສ່ວນ

- ການໃຊ້ງານ segment ຮ່ວມກັນໃນລະບົບ Segmentation
  - ການໃຊ້ງານ Segment ຮ່ວມກັນຈະບໍ່ສິ້ນເປືອງເທົ່າກັບການໃຊ້ງານ page ຮ່ວມກັນ
    - ໃນລະບົບ Segmentation ໂຄງສ້າງຂໍ້ມູນອາດຈະຫຍໍ້ ຫຼື ຂະຫຍາຍ ໂດຍບໍ່ປ່ຽນແປງ ຂໍ້ມູນທີ່ພົວພັນກັບ Segment ຂອງໂຄງສ້າງຂໍ້ມູນທີ່ກຳລັງໃຊ້ງານຮ່ວມກັນຢູ່.
  - ສອງຂະບວນການໃຊ້ງານ Segment ຮ່ວມກັນເມື່ອຂໍ້ມູນໃນຕາຕະລາງການ ປ່ຽນ Segment ຂອງພວກມັນຊີ້ໄປຫາ Segment ດຽວກັນໃນໜ່ວຍ ຄວາມຈຳຫຼັກ.
  - ເຖິງແມ່ນວ່າການໃຊ້ງານ Segment ຮ່ວມກັນຈະໃຫ້ຜົນປະໂຫຍດຢ່າງ ຊັດເຈັນ, ແຕ່ມັນກໍ່ມີຄວາມສ່ຽງຢູ່ນຳ
    - ຂະບວນການໜຶ່ງສາມາດປະຕິບັດງານອັນໃດໜຶ່ງໃນ Segment ໃດໜຶ່ງທີ່ມີຜົນກະທົບ ໃນທາງທີ່ບໍ່ດີຕໍ່ຂະບວນການອື່ນທີ່ໃຊ້ງານ Segment ນັ້ນຮ່ວມກັບມັນແບບຕັ້ງໃຈ ຫຼື ບໍ່ຕັ້ງໃຈ.

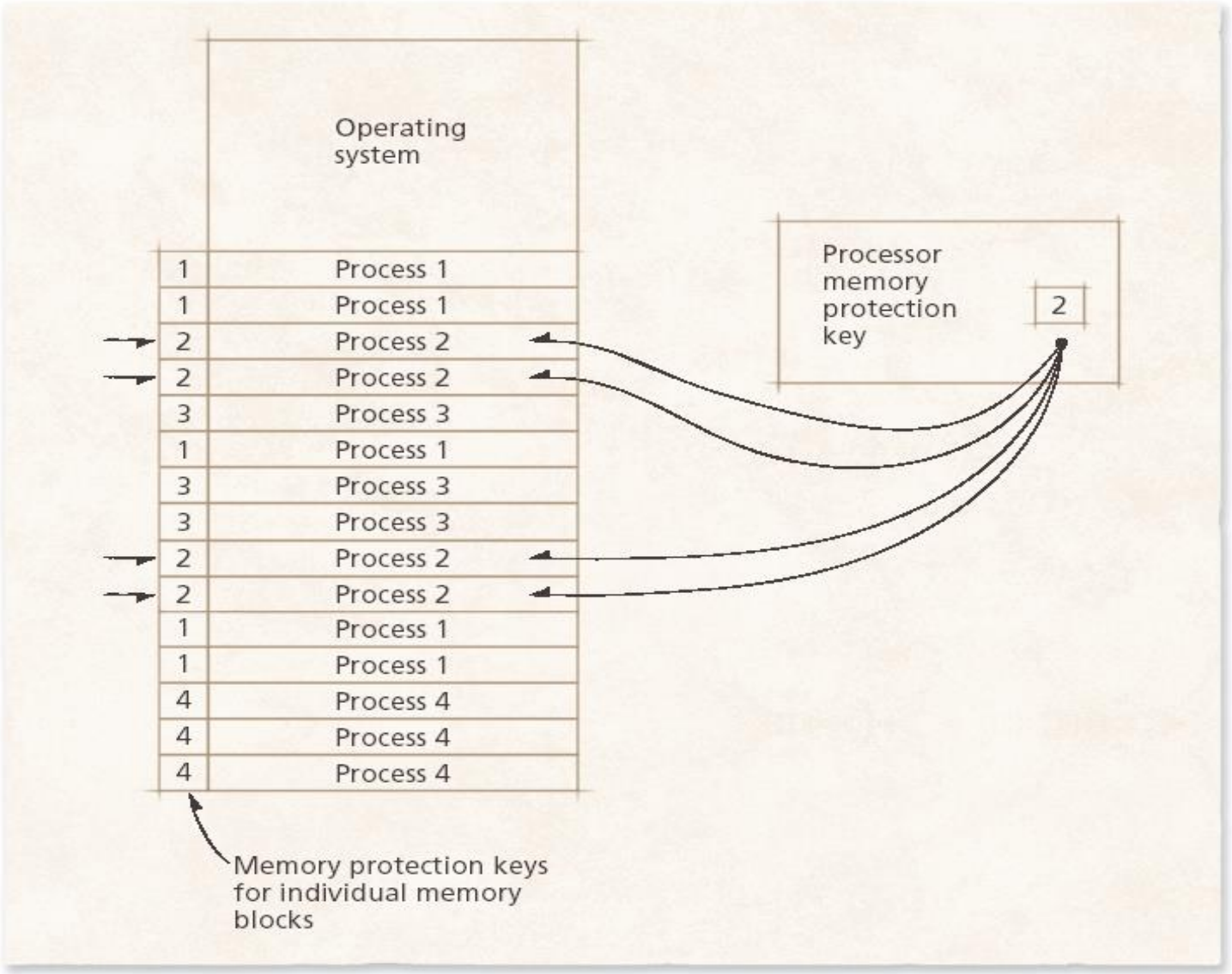
ภาพแบ่งส่วน


Process 1's segment table
Main memory
Segment shared by processes 1 and 2
Process 2's segment table

ການແບ່ງສ່ວນ

- ການປ້ອງກັນ ແລະ ການຄວບຄຸມການເຂົ້າໃຊ້ segment ໃນລະບົບ Segmentation
  - ຮູບແບບໜຶ່ງໃນການສ້າງການປ້ອງກັນໜ່ວຍຄວາມຈຳໃນລະບົບ Segmentation ແມ່ນການໃຊ້ key ປ້ອງກັນໜ່ວຍຄວາມຈຳ
  - ລະບົບປະຕິບັດການໃຊ້ key ປ້ອງກັນ ດັ່ງນີ້:
    - 1) ໃນເວລາຂະບວນການຖືກສັບປ່ຽນເຂົ້າອອກ, ລະບົບປະຕິບັດການໂລດ key ປ້ອງກັນຂອງຂະບວນການນັ້ນໄປເກັບໄວ້ໃນ Register ໃດໜຶ່ງຂອງໜ່ວຍປະມວນຜົນ.
    - 2) ເມື່ອຂະບວນການນັ້ນອ້າງອີງຫາ Segment ໃດໜຶ່ງ, ໜ່ວຍປະມວນຜົນຈະກວດສອບ key ປ້ອງກັນຂອງ block ທີ່ບັນຈຸຂໍ້ມູນທີ່ຖືກອ້າງອີງ.
    - 3) ຖ້າວ່າ key ປ້ອງກັນສຳຫຼັບຂະບວນການ ແລະ block ທີ່ຂໍຮ້ອງນັ້ນກົງກັນ ຂະບວນການນັ້ນສາມາດເຂົ້າໃຊ້ງານ Segment ນັ້ນໄດ້.

ການແບ່ງສ່ວນ


Operating system
1 Process 1
1 Process 1
2 Process 2
2 Process 2
3 Process 3
1 Process 1
3 Process 3
3 Process 3
2 Process 2
2 Process 2
1 Process 1
1 Process 1
4 Process 4
4 Process 4
4 Process 4
Processor memory protection key
2
Memory protection keys for individual memory blocks

ການແບ່ງສ່ວນ

- ການປ້ອງກັນ ແລະ ການຄວບຄຸມການເຂົ້າໃຊ້ segment ໃນລະບົບ Segmentation
  - ຮູບແບບທີ່ໃຊ້ທົ່ວໄປແມ່ນການໃຊ້ບິດປ້ອງກັນທີ່ກຳນົດວ່າຂະບວນການໃດໜຶ່ງສາມາດ Read, Write, Execute code ຫຼື Append ເຂົ້າກັບ Segment ນັ້ນ

Type of access	Abbreviation	Description
Read	R	This segment may be read.
Write	W	This segment may be modified.
Execute	E	This segment may be executed.
Append	A	This segment may have information added to its end.

ການແບ່ງສ່ວນ

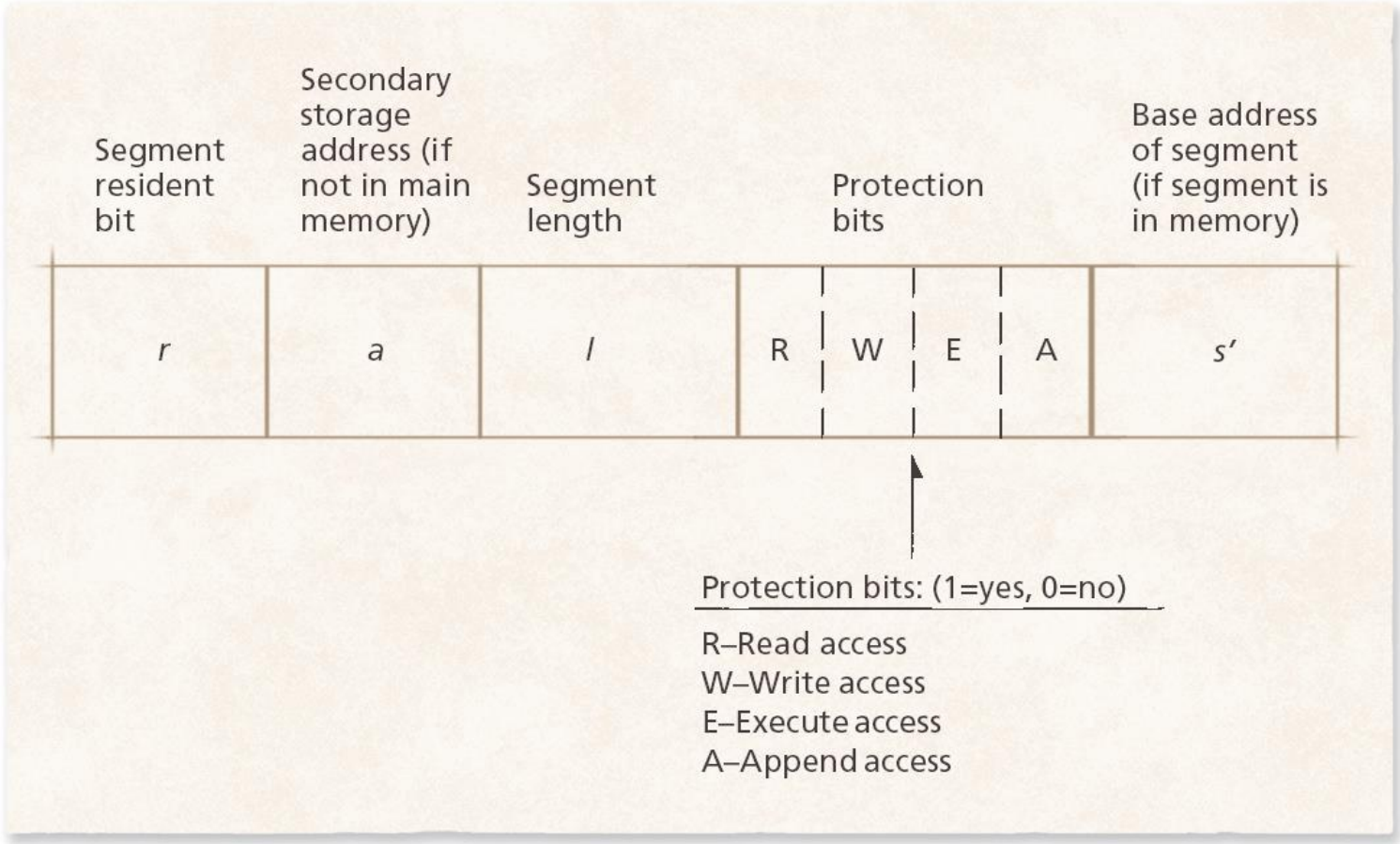
Mode	Read	Write	Execute	Description	Application
Mode 0	No	No	No	No access permitted	Security.
Mode 1	No	No	Yes	Execute only	A segment made available to processes that cannot modify it or copy it, but that can run it.
Mode 2	No	Yes	No	Write only	These possibilities are not useful, because granting write access without read access is impractical.
Mode 3	No	Yes	Yes	Write/execute but cannot be read	
Mode 4	Yes	No	No	Read only	Information retrieval.
Mode 5	Yes	No	Yes	Read/execute	A program can be copied or executed but cannot be modified.
Mode 6	Yes	Yes	No	Read/write but no execution	Protects data from an erroneous attempt to execute it.
Mode 7	Yes	Yes	Yes	Unrestricted access	This access is granted to trusted users.

ການແບ່ງສ່ວນ

- ການປ້ອງກັນ ແລະ ການຄວບຄຸມການເຂົ້າໃຊ້ segment ໃນລະບົບ Segmentation
  - ບິດປ້ອງກັນໄດ້ຖືກເພີ່ມເຂົ້າໄວ້ໃນຕາຕະລາງປ່ຽນ segment ແລະ ມັນຈະຖືກກວດສອບເມື່ອຂະບວນການໃດໜຶ່ງອ້າງອີງທີ່ຢູ່ໃດໜຶ່ງ
    - ເມື່ອ segment ນັ້ນບໍ່ຢູ່ໃນໜ່ວຍຄວາມຈຳຫຼັກ ລະບົບຈະສ້າງຂໍ້ຜິດພາດບໍ່ເຫັນ segment
    - ເມື່ອ d>1 ລະບົບຈະສ້າງຂໍ້ຜິດພາດທີ່ມີຂະໜາດເກີນ
    - ເມື່ອບໍ່ອະນຸຍາດໃຫ້ປະຕິບັດອັນໃດໜຶ່ງໃນບິດປ້ອງກັນ ລະບົບຈະສ້າງຂໍ້ຜິດພາດປ້ອງກັນ segment

ການແບ່ງສ່ວນ

- ...ะบิบ...มจะ



ລະບົບການແບ່ງໜ້າ ແລະ ການແບ່ງສ່ວນ

- ເປັນການປະສົມປະສານລະຫວ່າງ Paging ແລະ Segmentation
- ບັນດາ Segment ຈະຖືກສ້າງຈາກຫຼາຍ Page
- ທຸກ Page ຂອງ Segment ໃດໜຶ່ງບໍ່ຈຳເປັນຕ້ອງຢູ່ໃນໜ່ວຍຄວາມຈຳຫຼັກພ້ອມກັນໃນເວລາໃດໜຶ່ງ
- ບັນດາ Page ທີ່ເກັບຕໍ່ເນື່ອງກັນໃນໜ່ວຍຄວາມຈຳທຽມແມ່ນໃດກໍ່ບໍ່ຈຳເປັນຕ້ອງເກັບຕໍ່ເນື່ອງກັນແນວນັ້ນໃນໜ່ວຍຄວາມຈຳຫຼັກ

ລະບົບການແບ່ງໜ້າ ແລະ ການແບ່ງສ່ວນ

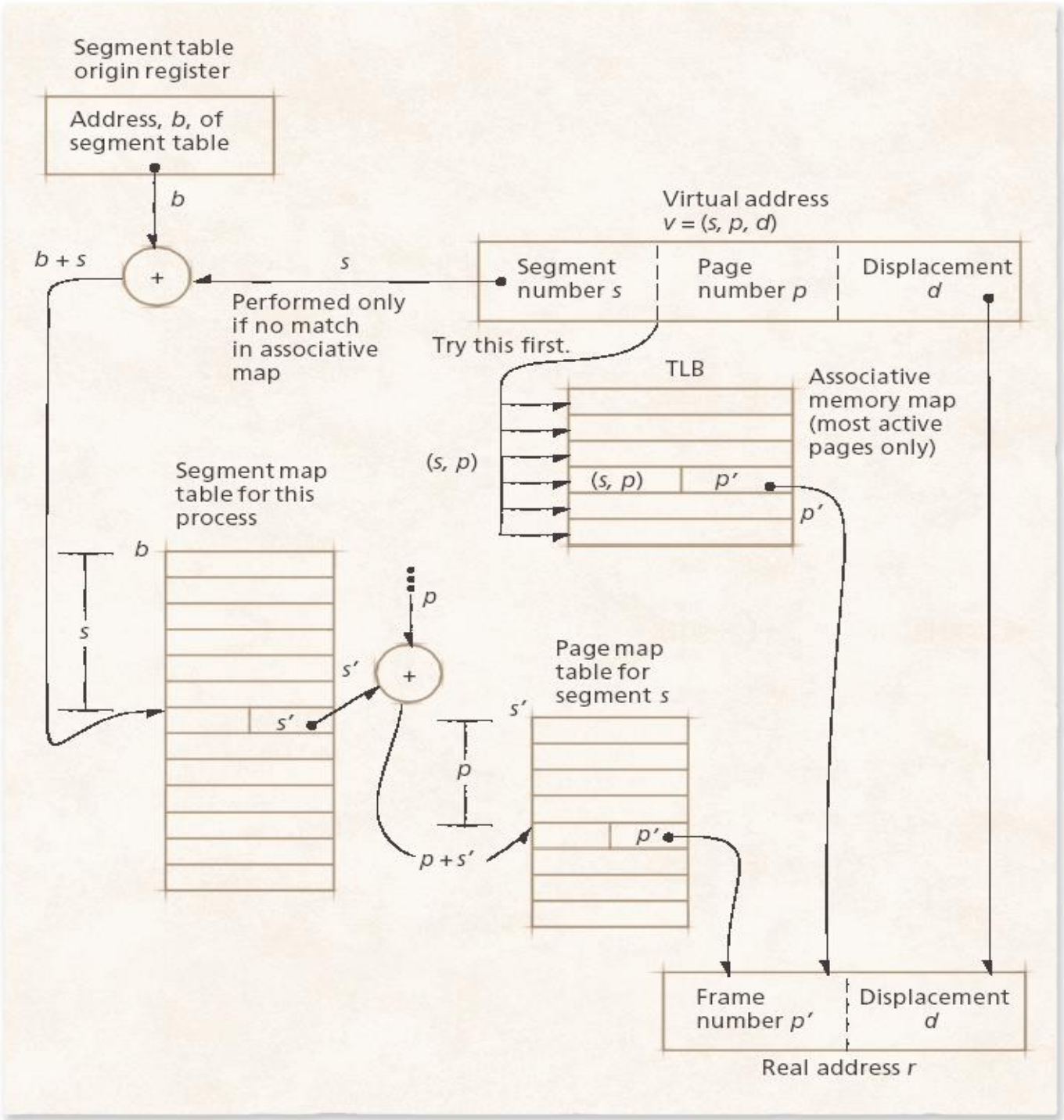
- ທີ່ຢູ່ຂອງໜ່ວຍຄວາມຈຳທຽມໃດໜຶ່ງແມ່ນສ້າງຂຶ້ນມາຈາກຕົວປະສານທີ່ມີສາມຕົວປ່ຽນ, v = (s, p, d), ເຊິ່ງໃນນັ້ນ
  - s ແມ່ນໝາຍເລກ Segment,
  - p ແມ່ນໝາຍ Page ພາຍໃນ Segment ແລະ
  - d ເລກລຳດັບຂອງຂໍ້ມູນໃນແຕ່ລະ Page ທີ່ຖືກອ້າງອີງ

ລະບົບການແບ່ງໜ້າ ແລະ ການແບ່ງສ່ວນ

- ການປ່ຽນທີ່ຢູ່ແບບບໍ່ຕາຍຕົວໃນລະບົບ Segmentation/ Paging
  - ຂະບວນການໃດໜຶ່ງຈະອ້າງອີງຫາທີ່ຢູ່ທຽມ v = (s, p, d)
  - DAT ຈະບອກທີ່ຢູ່ເລີ່ມຕົ້ນຂອງຕາຕະລາງປ່ຽນໝາຍເລກ Segment ຂອງຂະບວນການ, b, ໃສ່ທີ່ຢູ່ທີ່ຖືກອ້າງອີງ s
  - $b + s$ ຈະເປັນທີ່ຢູ່ໃນໜ່ວຍຄວມຈຳຫຼັກຂອງຕາຕະລາງການປ່ຽນ Segment ສຳຫຼັບ Segment s
  - ຕາຕະລາງການປ່ຽນ Segment ໄດ້ເກັບທີ່ຢູ່ເລີ່ມຕົ້ນຂອງຕາຕະລາງເລກໜ້າ s'
  - ໝາຍເລກໜ້າທີ່ຖືກອ້າງອີງ p ຈະຖືກບອກໃສ່ກັບ s' ເພື່ອກຳນົດທີ່ຕັ້ງ PTE ສຳຫຼັບ Page p ທີ່ເກັບ page frame p' ເອົາໄວ້
  - ລະບົບຈະເຊື່ອມຕໍ່ p' ໃສ່ກັບ d ເປັນທີ່ຢູ່ຈິງໃນໜ່ວຍຄວາມຈຳຫຼັກ

ละ

- ทา


Segment table origin register
Address, b, of segment table
b
b + s
+
s
Performed only if no match in associative map
Virtual address v = (s, p, d)
Segment number s
Page number p
Displacement d
Try this first.
TLB
Associative memory map (most active pages only)
(s, p)
p′
p′
Segment map table for this process
b
s
s′
s′
p
+
Page map table for segment s
s′
p
p′
p + s′
Frame number p′
Displacement d
Real address r

ยุดທະສາດໃນການປ່ຽນແທນ Pages

- ການປ່ຽນແທນ page ແບບສຸ່ມ (Random page replacement)
  - ເປັນຍຸດທະສາດທີ່ສ້າງໄດ້ງ່າຍ ແລະ ບໍ່ສິ້ນເປືອງ.
  - ແຕ່ລະ page ໃນໜ່ວຍຄວາມຈຳຫຼັກມີໂອກາດທີ່ຈະຖືກປ່ຽນແທນອອກໄປເທົ່າກັນ.
  - ບັນຫາໜຶ່ງທີ່ເກີດຂຶ້ນໃນວິທີນີ້ແມ່ນມັນອາດຈະໄປປ່ຽນແທນ page ທີ່ຈະຖືກເອີ້ນໃຊ້ຕໍ່ໄປນີ້ ເຊິ່ງຕ້ອງເສຍເວລາໄປເອີ້ນມັນກັບມາອີກ.
  - ຜົນປະໂຫຍດຂອງວິທີນີ້ແມ່ນສາມາດຕັດສິນໃຈປ່ຽນແທນໄດ້ໄວ ແລະ ຍຸດຕິທຳ

ยุดທະສາດໃນການປ່ຽນແທນ Pages

- ການປ່ຽນແທນ page ທີ່ເຂົ້າມາກ່ອນ (First-in-first-out page replacement)
  - ยุดทะสาดดั่งก่าอจะเลือกปรับแทน page ที่เข้ามาอยู่ในหน่วยความจำหลักดีนที่สุด.
  - ละบบจะบันทึกลำดับของ page ที่เข้ามาในหน่วยความจำหลัก
  - เป็นไปได้ที่ page เข้ามาก่อน จะเป็น page ที่ถูกเอิ้นใช้เป็นประจำ
  - สร้างขึ้นมาข้อมข้างบ่สิ้นเปืองโดยใช้ถือพรงอันดรอ
  - บ่เหมาะสมในการใช้งานตือจิงในเกือบทุกละบบ
  - เป็นพื้นถานให้แก่การสร้างยุดทะสาดการปรับแทนหลายประเพด

ยุดທະສາດໃນການປ່ຽນແທນ Pages

- ການປ່ຽນແທນ page ທີ່ບໍ່ໄດ້ໃຊ້ດົນແລ້ວ (Least-recently-used page replacement)
  - ມັນຈະປ່ຽນແທນ page ທີ່ຢູ່ໃນໜ່ວຍຄວາມຈຳດົນແລ້ວທີ່ບໍ່ຖືກອ້າງອີງ
  - ສາມາດໃຫ້ປະສິດທິພາບດີກວ່າ ການປ່ຽນແທນ page ທີ່ເຂົ້າມາກ່ອນ, ແຕ່ມັນກໍ່ເຮັດໃຫ້ມີການສິ້ນເປືອງເວລາໃນການເຮັດວຽກຂອງລະບົບ
  - ການເອົາວິທີການດັ່ງກ່າວມາໃຊ້ງານຈະຕ້ອງໄດ້ລະມັດລະວັງສະເໝີໃນການອອກແບບລະບົບປະຕິບັດການ ມັນອາດຈະເກີດຄວາມຜິດພາດໃນສະພາບການໃດໜຶ່ງ
    - page ທີ່ບໍ່ຖືກອ້າງອີງໃນຫວ່າງມໍ່ໆນີ້ອາດຈະເປັນ page ຕໍ່ໄປທີ່ຖືກອ້າງອີງດ້ວຍໂປຣແກຣມທີ່ເຮັດວຽກເປັນແບບລຳດັບຊັ້ນໃນການເຮັດວຽກເປັນຮອບວຽນ (Loop) ທີ່ອ້າງອີງຫຼາຍໆ page

ยุดທະສາດໃນການປ່ຽນແທນ Pages

- ການປ່ຽນແທນ page ທີ່ບໍ່ຖືກເອີ້ນໃຊ້ເປັນປະຈຳ (Least-Frequently-used page replacement)
  - ໃຊ້ວິທີການຕັດສິນໃຈໃນການປ່ຽນແທນໂດຍອີງຕາມຈຳນວນຄັ້ງທີ່ page ນັ້ນຖືກເອີ້ນໃຊ້ງານ
  - ລະບົບຈະປ່ຽນແທນ page ທີ່ບໍ່ຖືກເອີ້ນໃຊ້ເປັນປະຈຳ ຫຼື ຈຳນວນຄັ້ງໃນການອ້າງອີງໜ້ອຍ.
  - ວິທີການນີ້ຖືກສ້າງຂຶ້ນໂດຍໃຊ້ຕົວນັບຕົວໜຶ່ງທີ່ຖືກອັບເດດໃນແຕ່ລະຄັ້ງທີ່ page ນັ້ນຖືກອ້າງອີງ, ແຕ່ມັນອາດຈະເຮັດໃຫ້ເກີດການສິ້ນເປືອງຫຼາຍ
  - ອາດຈະເລືອກປ່ຽນແທນ page ທີ່ບໍ່ຖືກຕ້ອງແບບງ່າຍໆ.
    - ຕົວຢ່າງ page ທີ່ບໍ່ຖືກເອີ້ນໃຊ້ເປັນປະຈຳອາດຈະເປັນ page ທີ່ຫາກໍ່ໂລດເຂົ້າມາໃໝ່ຍອາມຈຳໃນຫວ່າງມໍ່ໆນີ້.

ยุดທະສາດໃນການປ່ຽນແທນ Pages

- ການປ່ຽນແທນ page ທີ່ບໍ່ຖືກເອີ້ນໃຊ້ໃນຫວ່າງມໍ່ໆນີ້ (Not-used-recently page replacement)
  - ຄ້າຍຄືກັບການປ່ຽນແທນ page ທີ່ບໍ່ຖືກເອີ້ນໃຊ້ດົນແລ້ວ
  - ອີງຕາມແນວຄວາມຄິດທີ່ວ່າ page ທີ່ບໍ່ຖືກເອີ້ນໃຊ້ໃນຫວ່າງມໍ່ໆນີ້ແມ່ນຈະບໍ່ຖືກເອີ້ນໃຊ້ໃນອະນາຄົດອັນໃກ້ນີ້
  - ຖືກສ້າງໂດຍໃຊ້ 2 bit ຮາດແວຣ໌ຕໍ່ໜຶ່ງຂໍ້ມູນໃນຕາຕະລາງການປ່ຽນ page:
    - Referenced bit - ມັນຈະມີຄ່າເປັນ 0 ຖ້າວ່າ page ບໍ່ໄດ້ຖືກອ້າງອີງ ແລະ ມີຄ່າເປັນ 1 ຖ້າວ່າ page ນັ້ນໄດ້ຖືກອ້າງອີງ (ບາງຄັ້ງເອີ້ນວ່າ access bit)
    - Modified bit - ມັນມີຄ່າເປັນ 0 ຖ້າ page ນັ້ນບໍ່ໄດ້ຖືກປ່ຽນແປງ ແລະ ມີຄ່າເປັນ 1 ຖ້າວ່າ page ນັ້ນຖືກປ່ຽນແປງ

ยุดທະສາດໃນການປ່ຽນແທນ Pages

- ການປ່ຽນແທນ page ທີ່ບໍ່ຖືກເອີ້ນໃຊ້ໃນຫ່ວງມໍ່ໆນີ້ (Not-used-recently page replacement)
  - ເລີ່ມຕົ້ນ ລະບົບຈະກຳນົດຄ່າ Referenced bit ຂອງທຸກ page ໃຫ້ເປັນ 0. ເມື່ອຂະບວນການໃດໜຶ່ງອ້າງອີງຫາ page ໃດໜຶ່ງ ລະບົບຈະກຳນົດຄ່າ Referenced bit ຂອງ page ນັ້ນເປັນ 1.
  - Modified bit ຂອງທຸກໆ page ກໍ່ໄດ້ກຳນົດຄ່າໃຫ້ເປັນ 0 ເຊັ່ນດຽວກັນ ໃນເບື້ອງຕົ້ນ. ເມື່ອໃດກໍ່ຕາມທີ່ page ໃດໜຶ່ງຖືກປ່ຽນແປງ ລະບົບຈະກຳນົດຄ່າ Modified bit ຂອງ page ນັ້ນໃຫ້ເປັນ 1.
  - ເມື່ອລະບົບຕ້ອງການປ່ຽນແທນເອົາ page ໃດໜຶ່ງອອກ ຍຸດທະສາດດັ່ງກ່າວຈະພະຍາຍາມຄົ້ນຫາ page ທີ່ບໍ່ໄດ້ຖືກອ້າງອີງ, ຖ້າບໍ່ມີ page ດັ່ງກ່າວນັ້ນ ລະບົບຈະຕ້ອງປ່ຽນແທນ page ທີ່ຖືກອ້າງອີງ ແຕ່ມັນຈະຕ້ອງກວດສອບ Modified bit ກ່ອນວ່າ page ນັ້ນໄດ້ຖືກປ່ຽນແປງຫຼືບໍ່, ຖ້າບໍ່ຖືກປ່ຽນແປງ ລະບົບຈະເລືອກປ່ຽນແທນ page ດັ່ງກ່າວ