MENU

Project 3: Virtual Memory

1. Project Overview

This project implements a virtual memory (VM) system using segmentation and paging. The system manages the necessary segment and page tables (PTs) in a simulated main memory. It accepts virtual addresses (VAs) and translates them into physical addresses (PAs) according to the current contents of the segment and PTs. Two versions of the VM manager can be implemented: The simpler version assumes that the entire VA space is resident in physical memory (PM). The second supports demand paging.

The system is optionally extended with a translation look-aside buffer (TLB) to make the translation process more efficient.

2. Principles of VM

A *logical address space* is an abstraction of *PM*, consisting a sequence of imaginary memory locations, which are mapped onto PM locations for the purposes of execution. When the size of the logical address space is allowed to exceed the size of the actual physical space, the space is called a *virtual memory*.

2.1 Segmentation with Paging

The implementation of VM can employ segmentation, paging, or a combination of both.

With pure *segmentation*, the logical address space of a process is divided into variable-size blocks, each typically corresponding to a logical component of the process, such as the code, the static data, the stack, and the dynamic data. Each block is mapped into a contiguous portion of PM. A segment table (ST) is used to keep track of the starting addresses of all segments.

Each VA is divided into 2 components, (s, w), where s is the segment number and w is the offset within the segment. Thus, s is an offset into the segment table.

With pure *paging*, the logical address space of a process is divided into fixed-size blocks, called *pages*. The PA space is divided into fixed-size blocks called *page frames*. The frame size and the page size must be identical and thus any page may be mapped into any available frame. A PT is used to keep track of the page numbers belonging to the process.

Each VA is divided into 2 components, (p, w), where p is the page number and w is the offset within the page. Thus, p is an offset into the PT.

The main advantage of segmentation is that blocks correspond to logical entities of the process, which facilitates linking and sharing. The main advantage of paging is efficient memory management by eliminating the need to search for and maintain variable-size memory partitions.

To gain the advantages of both approaches, segmentation can be combined with paging in that each segment is divided into fixed-size pages, which do not have to be contiguous in PM.

Each entry of the ST points to a PT corresponding to one segment, and each entry of the PT points to one page comprising the address space of the process.

Each VA is then divided into 3 components, (s, p, w), where s is the segment number (offset into the segment table), p is the page number (offset into the PT), and w is the offset within the page.

2.2 Address Translation

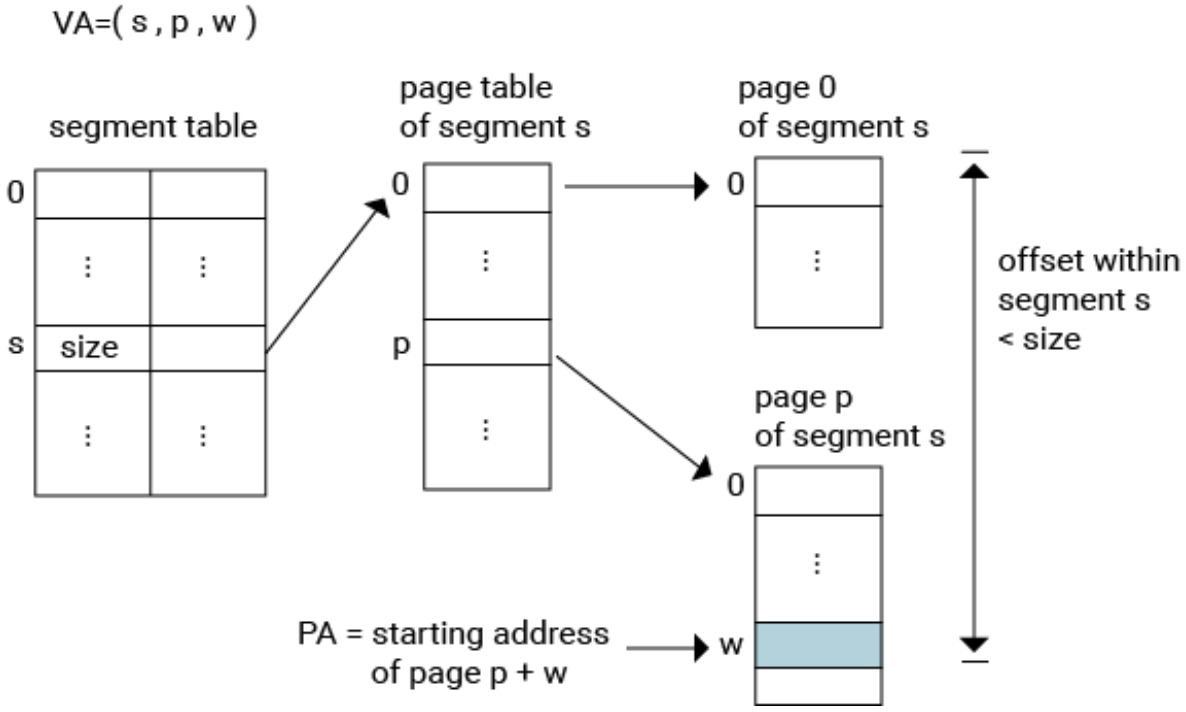
A VA is a nonnegative integer. The number of bits used to represent the VA determines the size of the VA space. For example, with a VA of 32 bits, 2^{32} addresses can be created.

The number of bits used to represent s, p, and w determine the sizes of the segment table, the PT, and each page, respectively.

A PA is also a nonnegative integer and the number of bits used to represent the PA determines the size of the PM.

The task of the VM manager is to translate VAs into corresponding PAs. The first step is to break the VA into the 3 components, s, p, and w, each of which becomes a separate integer.

As illustrated by the following diagram, the segment number s is used as an offset into the ST to find the corresponding PT. The page number p is used as an offset into the PT to find the corresponding page. The offset w is added to the starting address of the page to form the PA corresponding to the original VA.



Since segments have different sizes, only a subset of the possible VAs is valid. To prevent a process from accessing a location outside of a given segment s, the segment size is recorded in the ST and is checked against the offset within the segment s.

2.3 Demand Paging

If the size of the VM exceeds the size of the PM, then not all pages can be present in PM at all times, but must be loaded from a disk as needed. The principle is called *demand paging*. When a nonresident page is needed and no free frame is available, one of the resident pages must be removed and replaced by the new page. This principle is called *page replacement*.

Demand paging and page replacement applies also to PTs. Thus, the PT of a given segment may not be resident when a VA is being translated.

To keep track of which pages of a segment s are currently resident, an additional bit, called the *present bit* (sometimes called the resident bit), is associated with each entry of the PT of segment s. Similarly, each entry of an ST contains a present

bit to record the presence or absence of the corresponding PT. If the present bit is true, then the entry contains the frame number of the page or PT. If the present bit is false, then the entry contains the location of the page or PT on the disk.

Referencing a nonresident page or PT is called a *page fault*, which the memory manager must resolve by locating the missing page or PT on the disk, allocating a page frame, and loading the missing page or PT into the frame.

3. VM Without Demand Paging

3.1 VM Specification

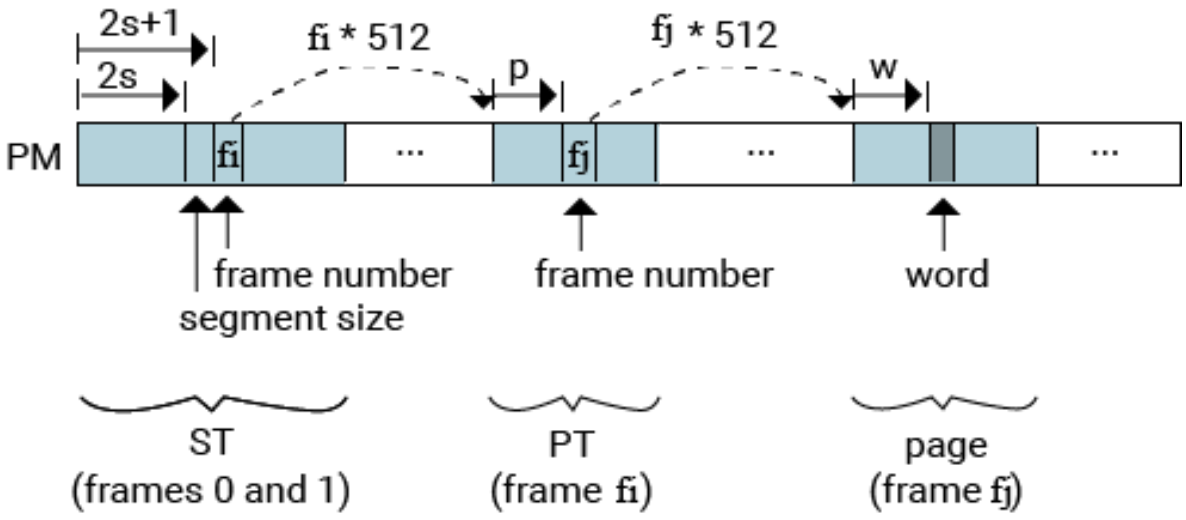
In the simplest case, we make the following assumptions:

- Memory is word-addressable, where each word is an integer.
- All tables and all pages are resident in memory, thus no page faults can occur.
- There is only a single process and hence only a single ST.
- Each entry of the ST consists of 2 integer components:
    - The size of the segment s (number of words)
    - The number of the page frame holding the PT of segment s

  If the segment does not exist, then both fields contain a 0.
- Each entry in a PT contains the number of the frame holding the corresponding page.
  If the page does not exist, then the entry contains a 0.
- A VA is an integer of 32 bits, divided into the 3 components: s, p, and w.
- Each component occupies 9 bits. Consequently:
    - The size of each page is 512 words (integers)
    - The size of each PT is also 512 words
    - The size of the ST is 1024 words (integers), since each entry occupies 2 integers
    - The leading 5 bits of the VA are unused

3.2 Representation of PM

- PM is represented as an array of integers, each corresponding to one addressable memory word. It is implemented as an array of 524,288 integers (= 2 MB).
- These are divided into 1024 frames of 512 words each. Consequently, the size of a PA is 19 bits.
- The figure below shows the organization of the VM in PM:



- Each entry of the ST requires 2 integers: one for the segment size and another for the frame number of the PT. Thus, the ST needs 2 frames and resides always in frames 0 and 1.
- Since each entry consists of 2 integers, the segment number s must be doubled to access the corresponding entry. PM[2s] refers to the size field of segment s and PM[2s+1] refers to the frame number, f_i, of the PT of segment s.
- To find the staring address of the PT, the frame number is multiplied by the frame size: PM[2s+1]*512
- Adding the page number p to the staring address of the PT yields the PT entry of page p: PM[2s+1]*512+p

- PM[PM[2s+1]*512+p] then contains the frame number, f_j, of the corresponding page p.
- To find the staring address of page p, the frame number is again multiplied by the frame size: PM[PM[2s+1]*512+p]*512
- Adding the offset w to the staring address of page p yields the final PA corresponding to the original VA: PM[PM[2s+1]*512+p]*512+w

3.1 Knowledge Check

Availability Start Date:

Oct 23, 2019 @ 10:00 PM PDT

Instructions:

NOTE: Please be sure to click the Save Answer button after typing/selecting your answer(s) then "I am Fisnished" to submit for grade.

Your Status:

BEGIN YOUR ATTEMPT (HTTPS://UCI.GRLCONTENT.COM/LAUNCH-ASSESSMENT/3945480?PAGE-ID=WEBCOM-VIEW-PAGE%2F5%2F7628%2F7624%3FPAGE-ID%3D422477).

3.3 Address Translation

The main task of the manager is to repeatedly accept VAs and attempt to translate them to the corresponding PAs. The first step is to break each VA into the 3 components s, p, and w, each represented as a separate integer.

The manager must also extract the component pw, represented as a single integer. This component is the offset into the segment s and must not exceed the segment size.

Deriving s, p, w, and pw from the VA

To extract the various VA components and represent each as a separate integer requires bit-manipulation operations. Assuming each component s, p, and w consists of 9 bits, the extraction can be accomplished as follows:

- s is extracted by right-shifting VA by 18 bits, which discards p and w.
- w is extracted by ANDing the VA with the 9-bit binary constant “1 1111 1111” (or 1FF in hexadecimal), which removes all bits other than the last 9 bits—the value of w.
- p is extracted by first right-shifting VA by 9 bits to discard w. The result is then ANDed with the binary constant “1 1111 1111,” which removes all bits other than the last 9 bits—the value of p.
- pw is extracted by ANDing the VA with the 18-bit binary constant “11 1111 1111 1111 1111” (or 3FFFF in hexadecimal), which removes the leading s component.

Translating VA to PA

Once all individual components of the VA have been extracted, the address translation performs the following steps:

- If pw ≥ PM[2s], then report error; VA is outside of the segment boundary
- Else PA = PM[PM[2s + 1]*512 + p]*512 + w

3.2 Knowledge Check

Availability Start Date:

Oct 23, 2019 @ 10:00 PM PDT

Instructions:

NOTE: Please be sure to click the Save Answer button after typing/selecting your answer(s) then "I am Fisnished" to submit for grade.

Your Status:

BEGIN YOUR ATTEMPT (HTTPS://UCI.GRLCONTENT.COM/LAUNCH-ASSESSMENT/3945482?PAGE-ID=WEBCOM-VIEW-PAGE%2F5%2F7628%2F7624%3FPAGE-ID%3D422477).

4. System Initialization and Execution

4.1 Initialization of the PM

The PM is initialized from a file, which specifies the frames of all PTs, the corresponding segment lengths, and the frame numbers of all pages. (ST always resides in frames 0 and 1.)

The file consists of 2 lines, each containing a series of integers:

Line 1: $s_1\ z_1\ f_1\ s_2\ z_2\ f_2\ \ldots\ s_n\ z_n\ f_n$

Line 2: $s_1\ p_1\ f_1\ s_2\ p_2\ f_2\ \ldots\ s_m\ p_m\ f_m$

Each triple $s_i\ z_i\ f_i$ on line 1 means that PT of segment s_i resides in frame f_i. and the length of segment s_i is z_i. Thus, line 1 defines the ST.

For example, 8 4000 3 means that the PT of segment 8 resides in frame 3 and the size of segment 8 is 4000. That is, the initialization sets PM[2*8] = PM[16] = 4000 and PM[2*8+1] = PM[17] = 3.

Each triple $s_j\ p_j\ f_j$ on line 2 means that the page p_j of segment s_j resides in frame f_j. Thus, line 2 defines the PTs.

For example, 8 5 8 means that page 5 of segment 8 resides in frame 8. That is, the initialization sets PM[PM[2*8+1]*512+5] = 8.

The actual pages of the VM are not represented explicitly in PM. The reason is that the manager only computes the PAs, but does not read or write the contents of the corresponding location PM[PA]. Thus, all pages contain zeros as the initial values of PM.

4.2 Executing VA Translations

Once the PM has been initialized, the system is ready to accept VAs and to attempt to translate them into PAs. The VAs are given in a second input file, which contains a series of integers, each representing one VA.

The program must read the file and attempt to translate each VA according to the rules given in Section 3.3. The result of each translation, which is either a corresponding PA or “error,” is to be written to an output file.

3.3 Knowledge Check

Availability Start Date:

Oct 23, 2019 @ 10:00 PM PDT

Instructions:

NOTE: Please be sure to click the Save Answer button after typing/selecting your answer(s) then "I am Fisnished" to submit for grade.

Your Status:

BEGIN YOUR ATTEMPT (HTTPS://UCI.GRLCONTENT.COM/LAUNCH-ASSESSMENT/3945484?PAGE-ID=WEBCOM-VIEW-PAGE%2F5%2F7628%2F7624%3FPAGE-ID%3D422477).

5. VM with Demand Paging

Under demand paging, not all pages or PTs are resident in PM, but are copied from a paging disk when accessed.

Pages can also be written back to the disk if no free frames are available, but this requires the implementation of a page replacement algorithm, which is beyond the scope of this project. Instead, we assume that PM is large enough to accommodate all address translation requests.

Several extensions must be added to the basic memory manager to support demand paging.

5.1 A Paging Disk

A physical disk consists of one or more rotating magnetic surfaces. Each surface consists of concentric tracks; each track is subdivided into sectors; and each sector consists of a fixed number of bytes. Modern disks typically hide the internal complexity by presenting the disk as a linear sequence of fixed-size blocks, which can be accessed one at a time using a block index, b.

In this project, the paging disk is emulated as a two-dimensional integer array, D[B][512], where B is the number of blocks and 512 is the block size (equal to the page size).

The disk may only be accessed one block at a time. The function *read_block(b, m)* copies the entire block D[b] into the PM frame starting at location PM[m].

All nonresident pages of the VM and PTs are kept on the paging disk.

5.2 Extended Contents of ST and PT

- To indicate that a PT is currently not resident in PM, the corresponding ST entry contains a negative number, −b, where the absolute value |−b| = b is the block number on the paging disk that contains the PT.
- Similarly, to indicate that a page is currently not resident in PM, the corresponding PT entry contains a negative number, −b, where the absolute value |−b| = b is the block number on the paging disk that contains the page.
- The sign bit in each ST or PT entry represents the present bit.

5.3 List of Free Frames

Since blocks may need to be moved to PM from the paging disk, the memory manager must keep track of which memory frames are free. A linked list can be used for that purpose.

5.4 VA Translation

Under demand paging, the address translation must implement additional steps:

Once all individual components of the VA have been extracted, the address translation performs the following steps:

- If pw ≥ PM[2s], then report error; VA is outside of the segment boundary
- If PM[2s + 1] < 0, then /* page fault: PT is not resident */
    - Allocate free frame f1 using list of free frames
    - Update list of free frames
    - Read disk block b = |PM[2s + 1]| into PM staring at location f2*512: read_block(b, f1*512)
    - PM[2s + 1] = f1 /* update ST entry */
- If PM[PM[2s + 1]*512 + p] < 0 /* page fault: page is not resident */
    - Allocate free frame f2 using list of free frames
    - Update list of free frames
    - Read disk block b = |PM[PM[2s + 1]*512 + p]| into PM staring at location f2*512: read_block(b, f2*512)
    - PM[PM[2s + 1]*512 + p] = f2 /* update PT entry */
- Return PA = PM[PM[2s + 1]*512 + p]*512 + w

3.4 Knowledge Check

Availability Start Date:

Oct 23, 2019 @ 10:00 PM PDT

Instructions:

NOTE: Please be sure to click the Save Answer button after typing/selecting your answer(s)

Your Status:

BEGIN YOUR ATTEMPT (HTTPS://UCI.GRLCONTENT.COM/LAUNCH-ASSESSMENT/3945486?PAGE-ID=WEBCOM-VIEW-PAGE%2F5%2F7628%2F7624%3FPAGE-ID%3D422477).

5.5 Initialization of Physical Memory

The file has the same format as before. However, the third component of each triple can be a positive or a negative integer. A positive integer, f_i, means that the corresponding PT or page resides in frame f_i. A negative integer, $-b_i$, means that the corresponding PT or page is currently not resident and must be loaded into a free frame from block b_i of the disk.

Line 1: $s_1\ z_1\ f_1/b_1\ s_2\ z_2\ f_2/b_2 \ldots s_n\ z_n\ f_n/b_n$

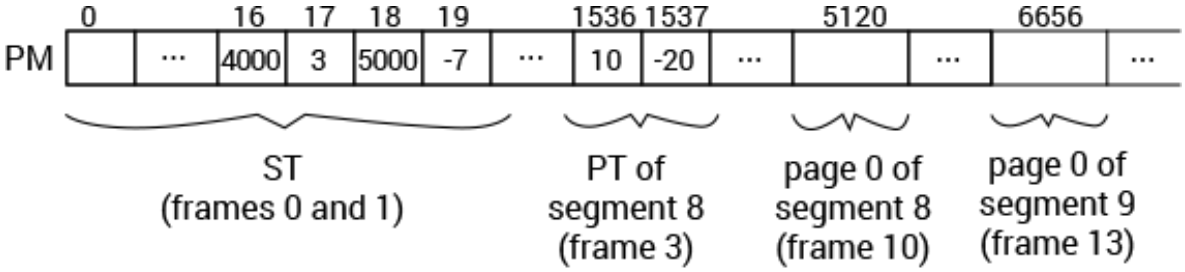
Line 2: $s_1\ p_1\ f_1/b_1\ s_2\ p_2\ f_2/b_2 \ldots s_m\ p_m\ f_m/b_n$

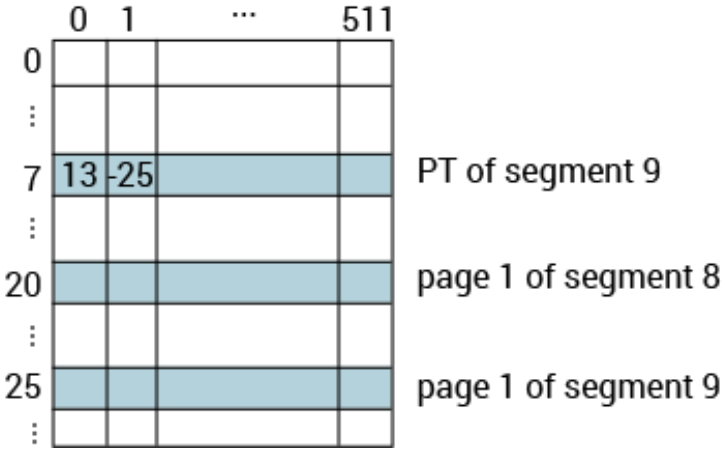
Example

The following initialization file results in a PM layout shown in the diagram below:

Line 1: 8 4000 3 9 5000 -7

Line 2: 8 0 10 8 1 -20 9 0 13 9 1 -25





- 8 4000 3 on line 1 means: PT of segment 8 resides in frame 3 and the size of segment 8 is 4000. Thus, PM[2*8] = 4000 and PM[2*8+1] = 3
- 9 5000 −7 on line 1 means: PT of segment 9 resides in disk block 7 and the size of segment 9 is 5000. Thus, PM[2*9] = 5000 and PM[2*9+1] = −7
- 8 0 10 on line 2 means: page 0 of segment 8 resides in frame 10.
  PT of segment 8 is resident in PM.
  Thus PM[PM[2*8+1]*512+0] = PM[PM[17]*512+0] = PM[3*512+0] = PM[1536] = 10.
- 8 1 −20 on line 2 means: page 1 of segment 8 resides in disk block 20.
  PT of segment 8 is resident in PM.
  Thus PM[PM[2*8+1]*512+1] = PM[PM[17]*512+1] = PM[3*512+1] = PM[1537] = −20.
- 9 0 13 on line 2 means: page 0 of segment 9 resides in frame 13.
  PT of segment 9 is on the disk.
  Thus D[|PM[2*9+1]|][0] = D[|PM[19]|][0] = D[|−7|][0] = D[7][0] = 13.
- 9 1 −25 on line 2 means: page 1 of segment 9 resides in disk block 25.
  PT of segment 9 is on the disk.
  Thus D[|PM[2*9+1]|][1] = D[|PM[19]|][1] = D[|−7|][0] = D[7][1] = −25.

As before, all pages are represented only implicitly and contain all zeros. In the example, this includes pages in frames 10 and 13 and in disk blocks 20 and 25.

6. The Translation Look-aside Buffer

6.1 Basic Principles

To speed up the address translation process, a TLB is employed. The TLB can be implemented for either version of the manager, that is, with or without demand paging.

A TLB is an associative memory, which maintains the set of the most recent address translations to avoid repeated access to the ST and PT. Each entry of the TLB contains the following fields:

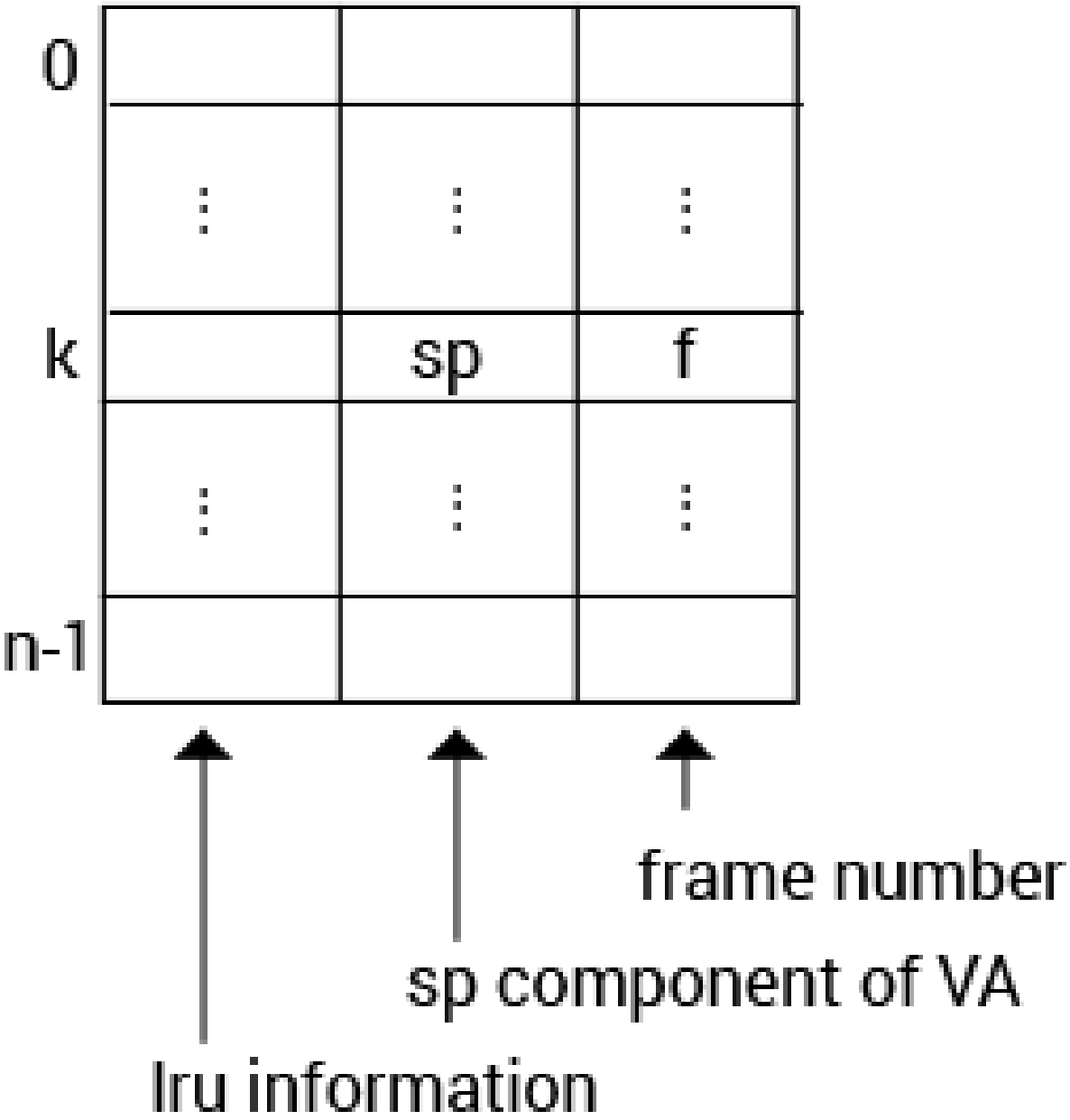
- The sp component of a VA, which refers to page p of segment s in VM
- The frame, f, where the page p currently resides

When a new VA is to be translated, the TLB is searched associatively for a match on sp. If a match is found (referred to as TLB hit), then the PA is formed by adding the offset, w, to the staring address of the frame f: PA = f*512+w. This bypasses the ST and PT, resulting in only a single memory access.

The TLB size is limited and thus not all translations can be maintained indefinitely. If no match is found for a given sp and no TLB entry is free, one of the existing entries must be replaced.

To exploit the principle of locality, the TLB always replaces the least recently used (LRU) entry. For that purpose, each entry contains a third field, which maintains the relevant LRU information. Each of the LRU fields contains an integer between 0 and $n - 1$, where n is the number of lines of the TLB. A 0 represents the LRU entry and $n - 1$ represents the most recently used entry.

The following diagram shows the layout of the TLB:



6.2 VA Translation with TLB

For each VM to be translated, the manager performs the following operations:

- Extract the components, s, p, w, pw, sp, each represented as an integer
- If pw ≥ PM[2s], then report error; VA is outside of the segment boundary
- If VA is valid, then search TLB for a match on sp
- If a match is found (TLB hit) in line k then:
    - Use the corresponding f from the TLB to form the PA, PA=f*512+w

- Update the LRU fields of line k as follows:
  - Decrement all LRU values that are greater than the LRU value of entry k
  - Set LRU of entry k to the maximum, $n - 1$

- If no match is found (TLB miss) then:
  - Resolve the VA as described in Section 5.4
  - Update TLB as follows:
    - Select line k with LRU=0 and set the LRU value to the maximum, $n - 1$
    - Replace the sp field of line k with the new sp value
    - Replace the f field of line k with PM[PM[2s+1]*512 + p]
    - Decrement all other LRU values by 1

- Output the PA or “error”
- Indicate also whether a miss or hit occurred in the TLB by outputting “m” or “h” following the PA

Example

Assume that the TLB consists of 4 lines with the contents as shown in diagram (a).

TLB

0	3	sp0	f0
1	1	sp1	f1
2	0	sp2	f2
3	2	sp3	f3

(a)

Changes after TBL hit

0	2		
1	3		
2	0		
3	1		

(b)

Changes after TLB miss

0	1		
1	2		
2	3	sp’	f1
3	0		

(c)

Diagram (b) shows the changes if a match for sp_1 is found in line 1. The LRU value of line 1 is set to the highest value 3. The values 2 and 3 are decrement to 1 and 2 respectively.

This sp_1 becomes the most recently accessed item and sp_2 remains the least recently accessed item.

Diagram (c) shows the changes if no match for a new value sp' is found. Line 2, which has the lowest LRU value, is selected for replacement. The LRU value of line 2 is set to the highest value 3. All other LRU values are decremented by 1. The original values of sp_2 and f_2 are replaced with the new values sp' and f'.

7. SUMMARY OF SPECIFIC TASKS

1. Design and implement a VM manager using segmentation and paging
    ~~a. Without demand paging, or~~
    b. With demand paging
2. Design and implement a TLB to speed up the address translation process
    a. Initialize the sp and f values to be empty, and the LRU field of line i to the value i
3. Design and implement a driver program that
    a. Initializes the system from a given input file
    b. Reads another input file and, for each VA, attempts to translate it to the corresponding PA; the result of each address translation are written into a new file
4. Create different initialization files and test the memory manager with different sequences of VAs

For questions or concerns regarding this online publication, please contact web support (https://uci.grlcontent.com/eform/submit/support-form).

Having troubles viewing something on the page, make sure you have the correct plug-ins (https://uci.grlcontent.com/systemRequirements).

View the Great River Learning Privacy Statement (https://uci.grlcontent.com/showPrivacyPolicy) | Terms and Conditions (https://uci.grlcontent.com/termsOfUsePage).