CSE-312
OPERATİNG SYSTEM
HW-3 part1

Refik Orkun ARSLAN
151044063

The Second Chance (SC)

.The referenced variable is equal to the second chance bit.

```
int replaceIndex = pt.queue.at(0), pageTableIndex = 0;
for (pageTableIndex = 0; pageTableIndex < pt.tableSize; ++pageTableIndex)
    if (replaceIndex == pt.table[pageTableIndex].page_index)
        break;

PageTableEntry ptReplace = pt.table[pageTableIndex];
virtualArray->pm->valids[ptReplace.pmpage_index]=1;

int validPMIndex = getValidpage_index();
virtualArray->pr.page_indexPM = validPMIndex;

auto it = pt.queue.begin();

if (ptReplace.referenced)
{
    replaceIndex = pt.queue.at(1);
    ptReplace.referenced=0;
    ++it;
}
for (pageTableIndex = 0; pageTableIndex < pt.tableSize; ++pageTableIndex)
    if (replaceIndex == pt.table[pageTableIndex].page_index)
        break;

pt.queue.erase(it);
pt.queue.push_back(page_index);
ptReplace.page_index=page_index;
ptReplace.pmpage_index=validPMIndex;

 writeDisk(ptReplace.pmpage_index);

 pt.table[pageTableIndex]= ptReplace;
```

.If the second chance bit is ONE, reset its second chance bit (to ZERO) and continue.

. If the second chance bit is ZERO, replace the page in that memory frame.
.The page to be changed has been found, deleted, and a new page has been added.
.New page write disk

Least-Recently-Used (LRU)

```
int replaceIndex = pt.queue.at(0), pageTableIndex = 0;
for (pageTableIndex = 0; pageTableIndex < pt.tableSize; ++pageTableIndex)
    if (replaceIndex == pt.table[pageTableIndex].page_index)
        break;
PageTableEntry ptReplace = pt.table[pageTableIndex];
virtualArray->pm->valids[ptReplace.pmpage_index]= 1;

int validPMIndex =getValidpage_index();
virtualArray->pr.page_indexPM = validPMIndex;

writeDisk(ptReplace.pmpage_index);

ptReplace.page_index=page_index;
ptReplace.pmpage_index=validPMIndex;

auto it = pt.queue.begin();
pt.queue.erase(it);
pt.queue.push_back(page_index);

pt.table[pageTableIndex]= ptReplace;
```

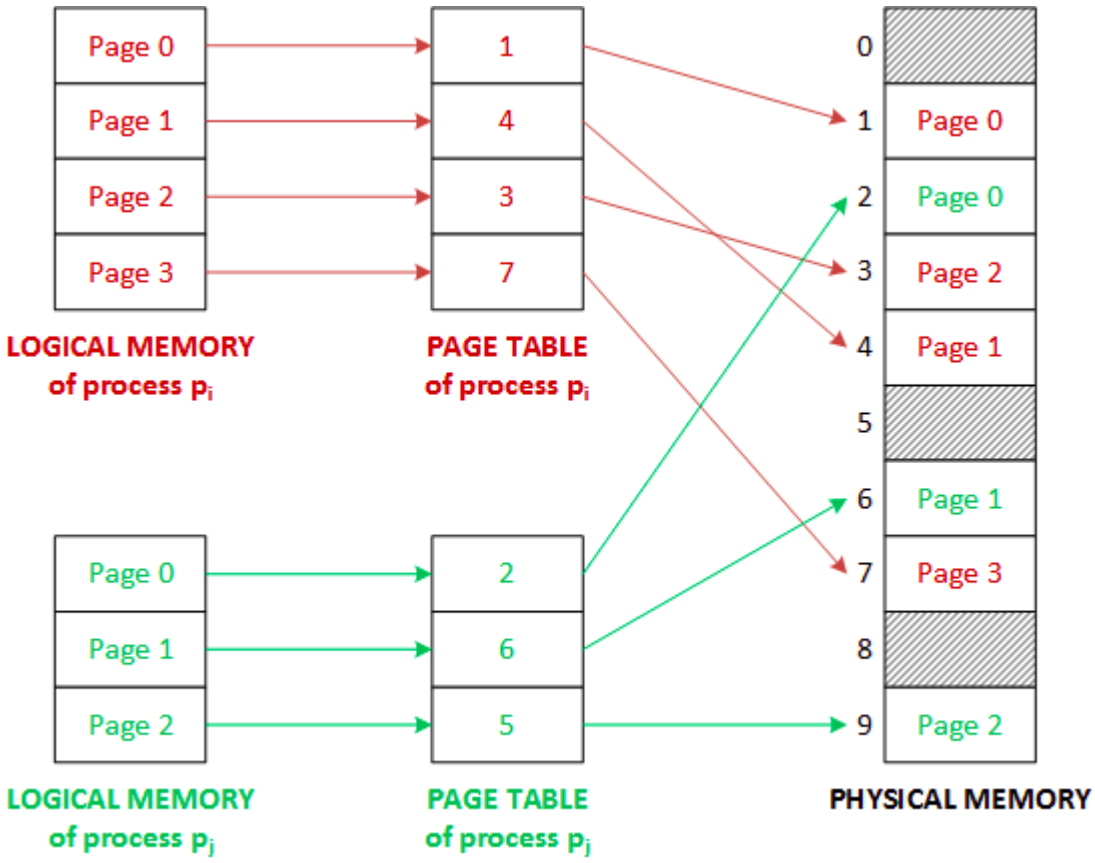
- .Whichever was used last will be replaced.
- .The last used page is found and replaced.
- .The most recently used page is kept in the queue so it was easy to reach.
- .Queues are based on the FIFO principle, So, 0.index This is the least used page.

Working Set Clock (WSClock)

```
int i = pt.queue.size()-1, pageTableIndex = 0, replaceIndex = 0;

for (; i >= 0; --i)
{
    replaceIndex = pt.queue.at(i);
    for (pageTableIndex = 0; pageTableIndex < pt.tableSize; ++pageTableIndex)
        if (replaceIndex == pt.table[pageTableIndex].page_index)
            break;
    if (pt.table[pageTableIndex].referenced)
    {
        pt.table[pageTableIndex].referenced=0;
    }
    else
    {
        virtualArray->pm->valids[pt.table[pageTableIndex].pmpage_index]= 1;
        int validPMIndex = getValidpage_index();
        virtualArray->pr.page_indexPM = validPMIndex;

        pt.table[pageTableIndex].page_index=page_index;
        pt.table[pageTableIndex].pmpage_index=validPMIndex;
        auto it = pt.queue.begin() + i;
        pt.queue.erase(it);
        pt.queue.push_back(page_index);

        replaceIndex = pt.queue.at(pt.queue.size()-1);
        for (pageTableIndex = 0; pageTableIndex < pt.tableSize; ++pageTableIndex)
            if (replaceIndex == pt.table[pageTableIndex].page_index)
                break;
        pt.table[pageTableIndex].referenced=1;
        break;
    }
}
```

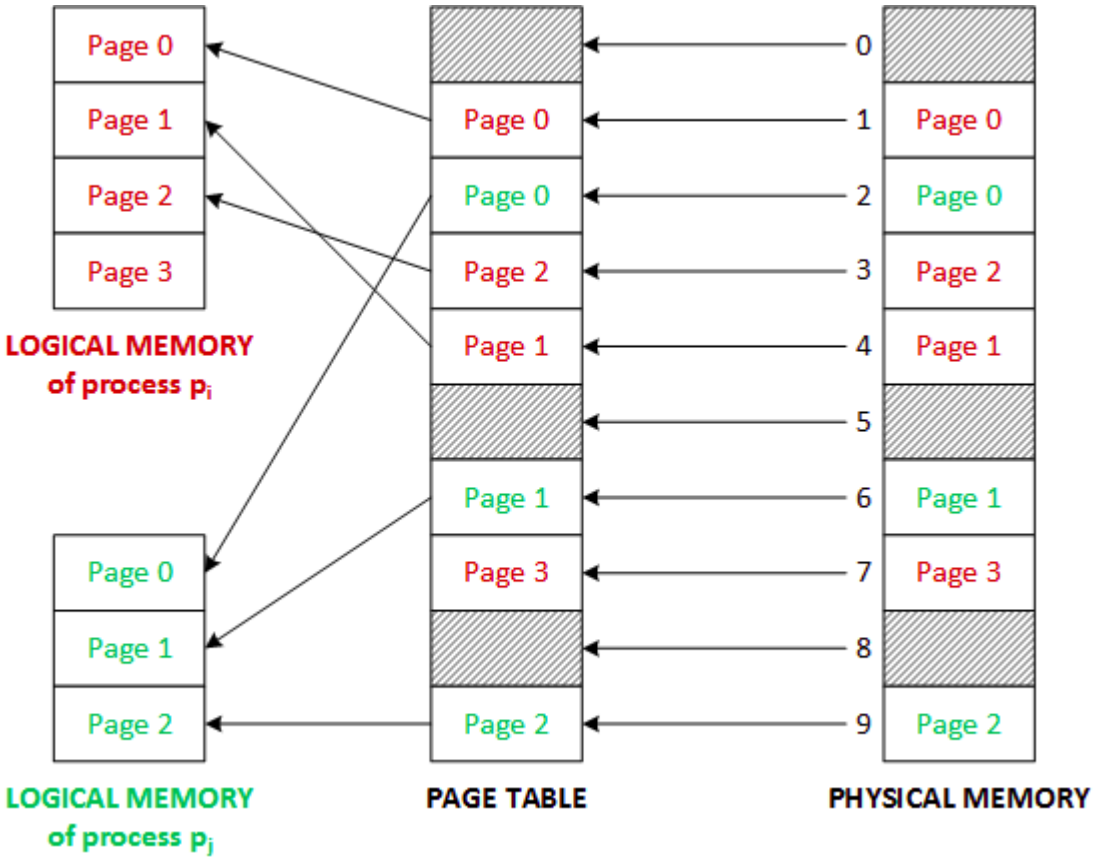
.Here the referenced is set to the R bit.
. If the R bit is set to 1, the page has been used during the current tick so it is not an ideal candidate to remove. The R bit is then set to 0, the hand advanced to the next page, and the algorithm repeated for that page.
.If the page pointed to has R = 0,it is not in the working set and a valid copy exists on the disk.That page call on disk.

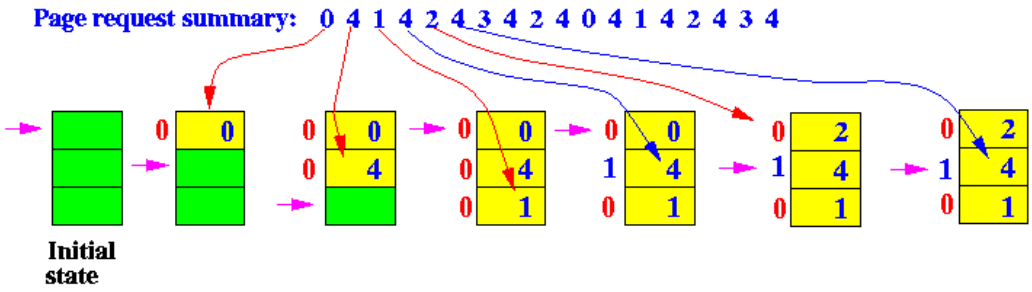
Regular Page Table

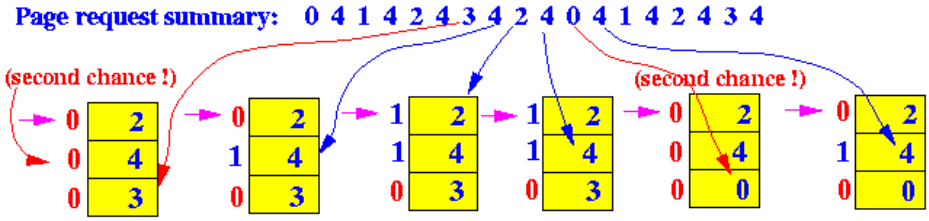


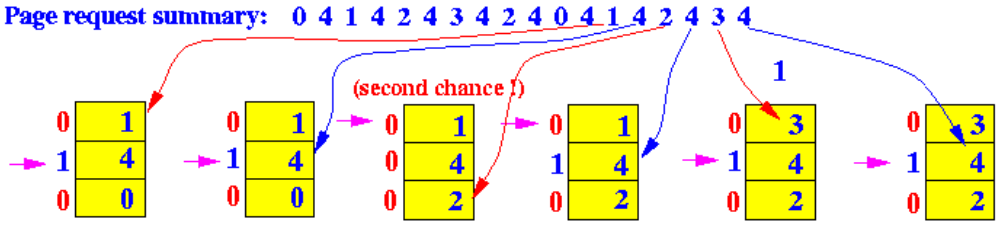
.A search is made within the page and accordingly we find whether it is a miss or a hit.
.When hit, referenced and modified is changed according to which page change happened, something is assigned to the queue.
.If miss, the page is retrieved from your physical memory.

Inverted Page Table




Page request summary: 0 4 1 4 2 4 3 4 2 4 0 4 1 4 2 4 3 4
Initial state


Page request summary: 0 4 1 4 2 4 3 4 2 4 0 4 1 4 2 4 3 4
(second chance !)
(second chance !)


Page request summary: 0 4 1 4 2 4 3 4 2 4 0 4 1 4 2 4 3 4
(second chance !)
1