Romil V. Shah
20008692

CS 520
Prof. Igor Faynberg

Homework Assignment #6
Submission Date: 11/22/2022

Q2. Write a program that implements the Buddy System algorithm for memory management. For input the program accepts 1) the amount of total available memory and 2) the sequence of integers with allocation (+) or return (-) requests. As output, the program must print the table showing the state of the memory (in the same way it is demonstrated in Lecture 6 after each allocation or return. Test the program with the following input parameters: Two megabytes of available memory and the following sequence: (A: +20K), (B: +35K), (C: +90K), (D: +40K), (E: +240K), (D: -40K), (A: -20K), (C: -90K), (B: -35K), (E: -240K). Turn in the listing of the program and the printout.

Sol.

The amount of total available memory: 2048

The sequence of integers with allocation (+) or return (-) requests:
(A: +20K), (B: +35K), (C: +90K), (D: +40K), (E: +240K), (D: -40K), (A: -20K), (C: -90K), (B: -35K), (E: -240K)

```
           | 0  31 |
A: +20K |A: 20

              | 0  31 | | 62  124 |
        |A: 20
B: +35K |A: 20    |B: 35

              | 0  31 | | 62  124 | | 125  249 |
        |A: 20
        |A: 20    |B: 35
C: +90K |A: 20    |B: 35     |C: 90

              | 0  31 | | 62  124 | | 125  249 | | 250  312 |
        |A: 20
        |A: 20    |B:35
        |A: 20    |B:35    |C:90
D: +40K |A: 20    |B:35    |C:90    |D:40

              | 0  31 | | 62  124 | | 125  249 | | 250  312 | | 500  749 |
         |A: 20
         |A: 20    |B:35
         |A: 20    |B:35    |C:90
         |A: 20    |B:35    |C:90    |D:40
E: +240K |A: 20    |B:35    |C:90    |D:40    |E:240
```

```
                    | 0  31 | | 62  124 | | 125  249 | | 250  312 | | 500  749 |
               |A: 20
               |A: 20     |B:35
               |A: 20     |B:35     |C:90
               |A: 20     |B:35     |C:90     |D:40
               |A: 20     |B:35     |C:90     |D:40     |E:240
D: -40K    |A: 20     |B:35     |C:90     |     |E:240

                    | 0  31 | | 62  124 | | 125  249 | | 250  312 | | 500  749 |
               |A: 20
               |A: 20     |B:35
               |A: 20     |B:35     |C:90
               |A: 20     |B:35     |C:90     |D:40
               |A: 20     |B:35     |C:90     |D:40     |E:240
               |A:20      |B:35     |C:90     |     |E:240
A: -20K    |     |B:35     |C:90     |     |E:240

                    | 0  31 | | 62  124 | | 125  249 | | 250  312 | | 500  749 |
               |A: 20
               |A: 20     |B:35
               |A: 20     |B:35     |C:90
               |A: 20     |B:35     |C:90     |D:40
               |A: 20     |B:35     |C:90     |D:40     |E:240
               |A:20      |B:35     |C:90     |     |E:240
               |     |B:35     |C:90     |     |E:240
C: -90K    |     |B:35     |     |     |E:240

                    | 0  31 | | 62  124 | | 125  249 | | 250  312 | | 500  749 |
               |A: 20
               |A: 20     |B:35
               |A: 20     |B:35     |C:90
               |A: 20     |B:35     |C:90     |D:40
               |A: 20     |B:35     |C:90     |D:40     |E:240
               |A:20      |B:35     |C:90     |     |E:240
               |     |B:35     |C:90     |     |E:240
               |     |B:35     |     |     |E:240
B:-35K     |     |     |     |     |E:240
```

```
                  | 0  31 | | 62  124 | | 125  249 | | 250  312 | | 500  749 |
              |A: 20
              |A: 20     |B:35
              |A: 20     |B:35     |C:90
              |A: 20     |B:35     |C:90     |D:40
              |A: 20     |B:35     |C:90     |D:40     |E:240
              |A:20      |B:35     |C:90     |     |E:240
              |    |B:35     |C:90     |     |E:240
              |    |B:35     |    |     |E:240
              |    |    |    |    |E:240
E: -240K  |    |    |    |    |
```

Q3. Read the Linux x.86 operational manual at Link and explain how Linux memory translation is done on x,86 processor.

Sol.

Q4. Below is an execution trace of a program fragment for a computer with 512-byte pages:

Load word 8118 (a value of a procedure input parameter) into register 0
Push register 0 onto the stack
Call a procedure at 4120 (the return address is stacked)
Remove one word (the actual parameter) from the stack and
Compare the actual parameter to constant 7
If equal, jump to 5000

The values of PC and SP are, respectively, 2020 and 10192. (The stack growth down toward 0.) Each instruction occupies four bytes. Produce the page reference string generated by this program.

Sol.

PC = 2020 = 2020 / 512 = on 3rd page

SP = 10192 = 10192 / 512 = on 19th page. This means data is referenced on 18th page.

1. Load word 8118 (a value of a procedure input parameter) into register 0

Sol. As PC in 2020, an instruction is on the 3rd page (2020 / 512) and data referenced on the 15th page (8118 / 512).

2. Push register 0 onto the stack

Sol. As each instruction occupies 4 bytes, PC is at 2024 (2020 + 4). This way, an instruction is on the 3rd page and SP is on the 18th page.

3. Call a procedure at 4120 (the return address is stacked)

Sol. PC is at 2028 (2024 + 4). An instruction is referenced on the 3rd page and SP is on the 18th page.

4. Remove one word (the actual parameter) from the stack and

Sol. After calling a procedure at 4120, PC is at 4120. This means, an instruction is on the 8th page and SP is on the 18th page.

5. Compare the actual parameter to constant 7

Sol. PC is at 4124 (4120 + 4). An instruction is on the 8th page.

6. If equal, jump to 5000

Sol. PC is at 4128 (4124 + 4). An instruction is on the 8th page and an address to be jumped on the 9th page.

The page reference string generated by this program is:
3, 15, 3 18, 3, 18, 8 18, 8, 8, 9

Q5.

1) Explain the value and drawbacks of “non-traditional” hardware architectures
for supporting virtual memory (i.e., inverted page table and TLB-only).

2) Using the inverted page table below,

0	2	Process1
1	3	Process2
2	1	Process1
3	0	Process2
4	0	Process1

give the physical page frame number corresponding to virtual page numbers 0, 1, and 2 of Process 1.

Sol.

1.

Values of an inverted page table:

- It contains only one record for each frame of memory.
- The size of the inverted page table is proportional to that physical memory.

Drawbacks of an inverted page table:

- An inverted page table is sorted by a physical address. That’s why when page reference occurs; it might need to search the whole table since the match is met.
- Since only one virtual page record for each frame of memory, one physical page can’t contain more than one shared virtual addresses. Otherwise, it will result in a page fault. It involves sharing memory issues.

Values of TLB:

- It is fast – lookup hardware cache.
- If TLB hit occurs then frame number becomes available immediately. This way search becomes fast.

Drawbacks of TLB:

- If TLB miss occurs then one has to add page number and frame number to the TLB.
- If TLB is already full of records then replacement policies should run to remove old entry.
- TLB will remember only one-page entry at a time if one entry of memory is used by two pages.

2.

Entry No.	Page No.	Process ID
0	2	Process 1
1	3	Process 2
2	1	Process 1
3	0	Process 2
4	0	Process 1

If match is found on i^{th} entry of inverted page table, then i will become a physical frame number.
For Process 1,

Virtual Page Number	Physical Page Frame Number
0	4
1	2
2	0

Q6. Prove that, with three-frame memory, the MRU will result in 3(l + 1) page faults, and LFU will result in 3[min (k, l) + 1] page faults, but the FIFO and LRU will result only in six-page faults on a reference string
w = (1, 2, 3)^k (4, 5, 6)^l.

Sol.
For reference, string = 1 4 2 5 3 6 (as k = 1 and l = 1)

MRU = 3(l + 1) = 3(1 + 1) = 3(2) = 6 Page Faults

		2	5	3	6
	4	4	4	4	4
1	1	1	1	1	1
Miss	Miss	Miss	Miss	Miss	Miss

Page Faults = 6

LFU = 3[min(k, l) + 1] = 3[1 + 1] = 3[2] = 6 Page Faults

		2	2	2	6
	4	4	4	3	3
1	1	1	5	5	5
Miss	Miss	Miss	Miss	Miss	Miss

Page Faults = 6

FIFO =

		2	2	2	6
	4	4	4	3	3
1	1	1	5	5	5
Miss	Miss	Miss	Miss	Miss	Miss

Page Faults = 6

LRU =

		2	2	2	6
	4	4	4	3	3
1	1	1	5	5	5
Miss	Miss	Miss	Miss	Miss	Miss

Page Faults = 6