

MIT WORLD PEACE UNIVERSITY

Operating Systems
Second Year B. Tech, Semester 3

MEMORY MANAGEMENT AND SIMULATION OF
PAGING ALGORITHMS

ASSIGNMENT 2
PRACTICAL REPORT

Prepared By
Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20
November 29, 2022

1 Code

```
1 #include <stdio.h>
2 #define MAX_FRAMES 10
3 #define MAX_PAGES 20
4 struct Frames
5 {
6     int page;
7     int insert_index;
8 } frames[MAX_FRAMES];
9
10 int pages[MAX_PAGES] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 3, 0, 0, 0, 0};
11
12 int frame_size = 4, no_of_pages = 14;
13 int hits = 0, faults = 0;
14 int page_search(int page)
15 {
16     for (int i = 0; i < frame_size; i++)
17     {
18         if (page == frames[i].page)
19         {
20             return i;
21         }
22     }
23     return -1;
24 }
25 void initialize_frame()
26 {
27     for (int i = 0; i < frame_size; i++)
28     {
29         frames[i].page = -1;
30         frames[i].insert_index = -1;
31     }
32 }
33 void display()
34 {
35     printf("Displaying frame: \n");
36     for (int i = 0; i < frame_size; i++)
37     {
38         printf("%d\n", frames[i].page);
39     }
40     printf("\n");
41 }
42 int where_to_insert()
43 {
44     int min = 1000;
45     int min_index = 0;
46     for (int i = 0; i < frame_size; i++)
47     {
48         if (frames[i].insert_index == -1)
49         {
50             return i;
51         }
52         else if (frames[i].insert_index <= min)
53         {
54             min = frames[i].insert_index;
55             min_index = i;
56         }
57     }
```

```
57     }
58     return min_index;
59 }
60 int lru()
61 {
62     for (int i = 0; i < no_of_pages; i++)
63     {
64         printf("Currently doing : %d\n\n", pages[i]);
65         int where = page_search(pages[i]);
66         if (where != -1)
67         {
68             printf("Hit\n");
69             hits++;
70             frames[where].insert_index = i;
71         }
72         else
73         {
74             printf("Miss\n");
75             faults++;
76             int temp = where_to_insert();
77             frames[temp].page = pages[i];
78             frames[temp].insert_index = i;
79         }
80         display();
81     }
82 }
83 int fifo()
84 {
85     for (int i = 0; i < no_of_pages; i++)
86     {
87         printf("Currently doing : %d\n\n", pages[i]);
88         if (page_search(pages[i]))
89         {
90             printf("Hit\n");
91             hits++;
92         }
93         else
94         {
95             printf("Miss\n");
96             faults++;
97             int temp = where_to_insert();
98             frames[temp].page = pages[i];
99             frames[temp].insert_index = i;
100         }
101         display();
102     }
103 }
104 // int
105 int main()
106 {
107     printf("Enter how many frames you have\n");
108     scanf("%d", &frame_size);
109     // printf("Enter how many Pages you have\n");
110     // scanf("%d", &no_of_pages);
111     // printf("Enter the Pages : \n");
112     // for (int i = 0; i < no_of_pages; i++)
113     // {
114     //     scanf("%d", pages[i]);
115     // }
```

```
116     printf("Executing First in First Out\n");
117     initialize_frame();
118     fifo();
119     printf("Hits: %d\n", hits);
120     printf("Faults: %d\n", faults);
121
122     hits = 0, faults = 0;
123     printf("Executing Least Recently Used\n");
124     initialize_frame();
125     lru();
126     printf("Hits: %d\n", hits);
127     printf("Faults: %d\n", faults);
128
129     return 0;
130 }
```

Listing 1: Code

2 Input and Output

```
1 Enter how many frames you have
2 3
3 Executing First in First Out
4 Currently doing : 7
5
6 Hit
7 Displaying frame:
8 -1
9 -1
10 -1
11
12 Currently doing : 0
13
14 Hit
15 Displaying frame:
16 -1
17 -1
18 -1
19
20 Currently doing : 1
21
22 Hit
23 Displaying frame:
24 -1
25 -1
26 -1
27
28 Currently doing : 2
29
30 Hit
31 Displaying frame:
32 -1
33 -1
34 -1
35
36 Currently doing : 0
37
38 Hit
39 Displaying frame:
```

```
40 -1
41 -1
42 -1
43
44 Currently doing : 3
45
46 Hit
47 Displaying frame:
48 -1
49 -1
50 -1
51
52 Currently doing : 0
53
54 Hit
55 Displaying frame:
56 -1
57 -1
58 -1
59
60 Currently doing : 4
61
62 Hit
63 Displaying frame:
64 -1
65 -1
66 -1
67
68 Currently doing : 2
69
70 Hit
71 Displaying frame:
72 -1
73 -1
74 -1
75
76 Currently doing : 3
77
78 Hit
79 Displaying frame:
80 -1
81 -1
82 -1
83
84 Currently doing : 0
85
86 Hit
87 Displaying frame:
88 -1
89 -1
90 -1
91
92 Currently doing : 3
93
94 Hit
95 Displaying frame:
96 -1
97 -1
98 -1
```

```
99
100 Currently doing : 2
101
102 Hit
103 Displaying frame:
104 -1
105 -1
106 -1
107
108 Currently doing : 3
109
110 Hit
111 Displaying frame:
112 -1
113 -1
114 -1
115
116 Hits: 14
117 Faults: 0
118 Executing Least Recently Used
119 Currently doing : 7
120
121 Miss
122 Displaying frame:
123 7
124 -1
125 -1
126
127 Currently doing : 0
128
129 Miss
130 Displaying frame:
131 7
132 0
133 -1
134
135 Currently doing : 1
136
137 Miss
138 Displaying frame:
139 7
140 0
141 1
142
143 Currently doing : 2
144
145 Miss
146 Displaying frame:
147 2
148 0
149 1
150
151 Currently doing : 0
152
153 Hit
154 Displaying frame:
155 2
156 0
157 1
```

```
158
159 Currently doing : 3
160
161 Miss
162 Displaying frame:
163 2
164 0
165 3
166
167 Currently doing : 0
168
169 Hit
170 Displaying frame:
171 2
172 0
173 3
174
175 Currently doing : 4
176
177 Miss
178 Displaying frame:
179 4
180 0
181 3
182
183 Currently doing : 2
184
185 Miss
186 Displaying frame:
187 4
188 0
189 2
190
191 Currently doing : 3
192
193 Miss
194 Displaying frame:
195 4
196 3
197 2
198
199 Currently doing : 0
200
201 Miss
202 Displaying frame:
203 0
204 3
205 2
206
207 Currently doing : 3
208
209 Hit
210 Displaying frame:
211 0
212 3
213 2
214
215 Currently doing : 2
216
```

```
217 Hit
218 Displaying frame:
219 0
220 3
221 2
222
223 Currently doing : 3
224
225 Hit
226 Displaying frame:
227 0
228 3
229 2
230
231 Hits: 5
232 Faults: 9
```

Listing 2: Input and Output