



KONYA

TEKNİK ÜNİVERSİTESİ

ÜNİVERSİTE	KONYA TEKNİK ÜNİVERSİTESİ
FAKÜLTE	MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ
BÖLÜM	YAZILIM MÜHENDİSLİĞİ
DERS -KONU	İŞLETİM SİSTEMLERİ – VİZE ÖDEVİ
OKUL NO	211229005
AD	ABDULMELİK
SOYAD	GÖYMEN
Github Link	https://github.com/gymgymm/thread



Soru 1

1'den 1.000.000 (1 milyon)'a kadar olan sayılardan oluşan bir ArrayList oluşturunuz. Ardından, bu ArrayList'teki 1milyon elemanı 250000 eleman olacak şekilde 4 eşit parçaya ayırınız. Bu 4 ayrı 250000 elemanlık ArrayList'ler içinde asal sayıları bulan 4 ayrı Thread tasarlayınız.

- a) 4 Thread bulduğu çift sayıları ortak bir ArrayList'e ekleyecektir.
- b) 4 Thread bulduğu tek sayıları ortak bir ArrayList'e ekleyecektir.
- c) 4 Thread bulduğu asal sayıları ortak bir ArrayList'e ekleyecektir.
- d) ArrayList'ler ilk oluşturulduklarında boş olacaklardır.

Soru 2

Amaç:

Bu ödevde, işletim sistemi üzerindeki sanal bellek yönetimini anlamak, sayfa değiştirme algoritmalarını incelemek, bellek hiyerarşisini kavramak ve bir uygulama üzerinden sanal bellek yönetimi problemlerini analiz etmek amaçlanmaktadır.

Görevler:

1- Sanal Bellek Kavramları:

Sanal bellek nedir, nasıl çalışır, işletim sistemi açısından avantajları ve dezavantajları nelerdir? Bu konuda kapsamlı bir literatür taraması yaparak kavramları açıklayınız.

2- Sayfa Değiştirme Algoritmaları:

FIFO, Optimal, LRU gibi sayfa değiştirme algoritmalarını araştırınız. Her bir algoritmanın avantajlarını ve dezavantajlarını belirleyerek karşılaştırmalı bir analiz yapınız.

3- Sanal Bellek Hiyerarşisi:

Bellek hiyerarşisinin nasıl işlediğini ve ana bellek (RAM), ikincil bellek (disk), sayfa tablosu gibi kavramların birbirleriyle nasıl etkileşimde bulunduğunu açıklayınız.

4- Sayfa Tablosu Analizi:

Sayfa tablosu nedir, nasıl çalışır? Sayfa tablosu oluşturulurken kullanılan teknikleri inceleyip avantaj ve dezavantajlarını belirtiniz.

5- Bellek Fragmentasyonu:

Bellek fragmentasyonu kavramını açıklayınız. İç ve dış fragmentasyon nedir? Sanal bellek yönetiminin bellek fragmentasyonu ile başa çıkma yöntemlerini araştırınız.

6- Sanal Bellek Yönetimi Uygulama:

İstedığınız bir işletim sistemi üzerinde belirli bir uygulamanın sanal bellek yönetimini inceleyiniz. Uygulama sırasında ortaya çıkabilecek durumları analiz ediniz.

Raporlama: Yukarıdaki görevleri içeren kapsamlı bir rapor hazırlayınız. Raporunuz, literatür taraması, kavramsal açıklamalar, analiz sonuçları ve öğrenimlerinizi içermelidir. Raporunuzu 12 punto Times New Roman yazı stili ve 1.5 satır aralığıyla hazırlayınız.

2022 – 2023 Akademik Yılı Yazılım Mühendisliği İşletim Sistemleri Vize Ödevi

Soru 1: 1'den 1.000.000 (1 milyon)'a kadar olan sayılardan oluşan bir ArrayList oluşturunuz. Ardından, bu ArrayList'teki 1milyon elemanı 25000 eleman olacak şekilde 4 eşit parçaya ayırınız. Bu 4 ayrı 25000 elemanlık ArrayList'ler içinde asal sayıları bulan 4 ayrı Thread tasarlayınız.

- 4 Thread bulduğu çift sayıları ortak bir ArrayList'e ekleyecektir.
- 4 Thread bulduğu tek sayıları ortak bir ArrayList'e ekleyecektir.
- 4 Thread bulduğu asal sayıları ortak bir ArrayList'e ekleyecektir.
- ArrayList'ler ilk oluşturulduklarında boş olacaklardır.

```
1  using System;
2  using System.Collections;
3  using System.Diagnostics;
4  using System.Threading;
5
6  0 başvuru
7  class Program
8  {
9      public static ArrayList arrayList = new ArrayList();
10     public static ArrayList tekarray = new ArrayList();
11     public static ArrayList çiftarray = new ArrayList();
12     public static ArrayList asalarray = new ArrayList();
13     public static ArrayList[] dividedArrayLists;
14     static ManualResetEvent event1 = new ManualResetEvent(false);
15     static ManualResetEvent event2 = new ManualResetEvent(false);
16     static ManualResetEvent event3 = new ManualResetEvent(false);
17     static ManualResetEvent event4 = new ManualResetEvent(false);
```

1. Aşama: Kodun 1. ve 5. Satırları arasında sistemi için gerekli kütüphaneler projeye dahil edilmiştir. Bizim projemizin ehemmiyeti açısından System.Threading kütüphanesi dikkat çekmektedir.

2. Aşama: Kodun 6. ve 17. Satırları arasında bulunan **ArrayList** tanımları ve elzem olan `static ManualResetEvent event1 = new ManualResetEvent(false);` tanımlaması bulunmaktadır. Peki bu tanımlama(lar) nedir:

- ManualResetEvent C# dilinde, System.Threading isim alanı altında yer alan bir senkronizasyon primitifidir. Bu nesne, birden fazla thread arasında sinyalizasyon ve bekleme mekanizmalarını sağlamak için kullanılır. ManualResetEvent özellikle, bir thread'in diğer thread'lerin belirli bir noktaya

ulaşmasını veya bir işlemi tamamlamasını beklemesi gereken durumlarda kullanışlıdır.

```
18 static void Main(string[] args)
19 {
20     // 1000000 öge ekleyerek doldur
21     for (int i = 0; i < 1000000; i++)
22     {
23         arrayList.Add(i);
24     }
25
26     // ArrayList'teki öğeleri yazdır
27     dividedArrayLists = DivideArrayList(arrayList, 4);
28     Stopwatch stopwatch = new Stopwatch();
29
30     stopwatch.Start();
31
32     Thread thread1 = new Thread(Fonk1);
33     thread1.Priority = ThreadPriority.Highest;
34
35     Thread thread2 = new Thread(Fonk2);
36     thread2.Priority = ThreadPriority.AboveNormal;
37
38     Thread thread3 = new Thread(Fonk3);
39     thread3.Priority = ThreadPriority.Normal;
40
41     Thread thread4 = new Thread(Fonk4);
42     thread4.Priority = ThreadPriority.BelowNormal;
43
44     thread1.Start();
45     thread2.Start();
46     thread3.Start();
47     thread4.Start();
48
49     WaitHandle.WaitAll(new WaitHandle[] { event1, event2, event3, event4 }); // Tüm olayların tamamlanmasını bekleyin
50
51     stopwatch.Stop();
52
53     Console.WriteLine($"Toplam thread çalışma süresi: {stopwatch.ElapsedMilliseconds} milisaniye");
54
55     Console.ReadLine();
56 }
```

4. Aşama: Kodun 18. satırından itibaren main kısmı başlamış bulunmaktadır.

- Kodun 20. satırında başlayıp 24. satırında biten **for** döngüde ise listeye 0 dan başlayıp 999.999 a kadar 1.000.000 adet ardışık sayıyı arrayList listesinde eklemektedir.
- arrayList listesine eklenen bu sayılar 27. satırda bulunan dividedarrayLists = DivideArrayList(arrayList, 4); ifadesi ise bir arrayList' i belli bir sayıda (4) alt gruba bölünmesi işlemini gerçekleştirmektedir. Bölünmesi gerçekleştirilen liste dividedarrayLists değişkenine atılıyor.
- Kodun 28. ve 30. satırlarında ise süre ölçmek için gerekli tanımlamalar yapıp süre sayacı başlatılmaktadır.
- Oluşturulan threadler ve atanan parametreler ışığında threadlere öncelik sırası atama işlemleri 32. ve 42. satırları arasında yapılmış ve işlemler şu şekilde gerçekleşmiştir:

```
Thread thread4 = new Thread(fonk4);  
thread4.Priority = ThreadPriority.BelowNormal;
```

BelowNormal komutu threadın çalışmasındaki öncelik sırasını belirler.

Öncelik sırası ve yapıları şu şekildedir:

1. **BelowNormal:** Öncelik seviyesi, "Normal"den biraz daha düşüktür. "BelowNormal" öncelikli bir thread, "Normal" öncelikli thread'lere göre daha az işlemci zamanı alır. Bu, daha az kritik işlemleri yürüten thread'ler için uygun olabilir.
2. **Normal:** Varsayılan thread önceliği seviyesidir. Çoğu thread bu seviyede başlatılır. "Normal" öncelik, thread'in işlemci zamanına ortalama bir erişim sağlar. Diğer öncelik seviyeleri, bu "normal" seviyeyle karşılaştırılarak değerlendirilir.
3. **AboveNormal:** Öncelik seviyesi, "Normal"den biraz daha yüksektir. "AboveNormal" öncelikli bir thread, "Normal" öncelikli thread'lere göre daha fazla işlemci zamanı alır. Daha önemli işlemleri yürüten thread'ler için bu seviye tercih edilebilir.
4. **Highest:** En yüksek thread önceliği seviyesidir. "Highest" öncelikli bir thread, diğer tüm öncelik seviyelerindeki thread'lere göre daha fazla işlemci zamanı alır. Çok kritik işlemleri yürüten thread'ler için bu seviye uygun olabilir.

Öncelik ayarlamak, özellikle kaynak yoğun uygulamalarda veya çoklu iş parçacığı kullanılan durumlarda önemlidir. Ancak, önceliklerin dikkatli bir şekilde kullanılması gerekir, çünkü yüksek öncelikli iş parçacıklarının aşırı kullanımı diğer iş parçacıklarının açlığa (starvation) uğramasına neden olabilir. Kodumda ise thread1'den thread4'e kadar öncelik sırası yüksekten düşüğe gidecek şekilde ayarlanıp koda dökülmüş ve threadler `thread1.Start();` komutuyla başlatılmıştır.

- 49. satırda ise `WaitHandle.WaitAll(new WaitHandle[] { event1, event2, event3, event4 });` komutu ile bütün işlemlerin bitmesi beklenmiştir.
- 51. Satırda da 28. satırda tanımlanıp 30. satırda başlatılan süre sayacı durdurulmuştur.
- 53. satırda console çıktıları verilmiştir.
- 55. satırda ise console uygulamasının bir tuşa basıncaya dek açık kalması için gerekli komut yazılmıştır.

KOD lock OLMAYAN ve lock OLAN ŞEKLİNDE 2' YE AYRILMAKTADIR

Lock Nedir: Çoklu-thread uygulamalarda, birden fazla thread'in aynı kaynağa (örneğin, bir veritabanına, bir dosyaya veya hafızadaki bir değişkene) eş zamanlı olarak erişmeye çalışması durumunda veri tutarsızlığı, yarış koşulları (race conditions) ve diğer senkronizasyon sorunları ortaya çıkabilir. "lock" ifadesi, bu tür problemleri önlemek için kullanılır.

Lock Nasıl Çalışır: C#'ta lock ifadesi, Monitor sınıfının bir kolay kullanım şeklidir. Lock bloğu, belirli bir nesne üzerinde kilit (lock) oluşturur. Bir thread kilitlediği zaman, diğer thread'ler bu nesnenin kilidini alana kadar beklemek zorunda kalır.

LOCK OLMAYAN KOD:

```

123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
265
```

- Bu aşamada her thread için bir fonksiyon oluşturulmuştur. Yani her thread sadece 1 tane fonksiyon ve kendi fonksiyonunu çalıştırmaktadır.
- Devamı ise asal sayı bulma şeklinde devam etmektedir.
- Bulunan asal sayılar `asalarray` listesine, tek sayılar `tekarray` listesinde, çift sayılar ise `çiftarray` listesine eklenmektedir.
- 128. satırda ise ilgili thread için başlatılan süre sayacı durdurulup ölçüm yapılıyor.

İşlemler neticesinde örnek konsol çıktısı şekildeki gibidir:

```

Thread 1 çalışma süresi: 11965 milisaniye
Thread 2 çalışma süresi: 34240 milisaniye
Thread 3 çalışma süresi: 50814 milisaniye
Thread 4 çalışma süresi: 66280 milisaniye
Toplam thread çalışma süresi: 66232 milisaniye
Tek sayı adeti: 499927
Çift sayı adeti: 499926
Asal sayı adeti: 78495
  
```

➤ Geliştirilen thread ile asal sayı bulma kodunun console çıktılar yukarıdaki bulunan fotoğraftaki gibidir. Burada süre çıktıları yanı sıra toplam sayı adeti 1.000.000 ‘dan düşük olduğunu gözlemlemekteyiz. Kod çalışma esnasında bir noktada(larda) listelerde bulunan sayıların da veri kaybı olmaktadır. Bu durumu bertaraf etmek için C# dilinde bulunan ve yukarıda bahsi geçen lock ifadesini kullanacağız.

LOCK OLAN KOD: Burada ise diğer kod yapısı aynı sadece fonksiyonlarda bulunan `asalarray`, `çiftarray`, `tekarray` listelerine ekleme bölgelerine lock ifadesini atacağız ki işlemler yapıldığında veri kaybı olmasın.

```

83 static void Fonk1()
84 {
85     Stopwatch stopwatch = new Stopwatch();
86     stopwatch.Start();
87     int uzunluk = dividedArrayLists[0].Count;
88     for (int i = 0; i < uzunluk; i++)
89     {
90         int count = 0;
91         for (int j = 2; j <= (int)dividedArrayLists[0][i] / 2; j++)
92         {
93             if ((int)dividedArrayLists[0][i] % j == 0)
94             {
95                 count++;
96                 break; // asal olmadığa anlaşılr anında çık
97             }
98         }
99
100         if (count == 0)
101         {
102             if ((int)dividedArrayLists[0][i] != 0 && (int)dividedArrayLists[0][i] != 1)
103             {
104                 lock (asalarray)
105                 {
106                     asalarray.Add((int)dividedArrayLists[0][i]);
107                 }
108             }
109         }
110
111         if ((int)dividedArrayLists[0][i] % 2 == 0)
112         {
113             lock (çiftarray)
114             {
115                 çiftarray.Add(dividedArrayLists[0][i]);
116             }
117         }
118         else
119         {
120             lock (tekarray)
121             {
122                 tekarray.Add(dividedArrayLists[0][i]);
123             }
124         }
125     }
126     stopwatch.Stop();
127     Console.WriteLine($"Thread 1 çalışma süresi: {stopwatch.ElapsedMilliseconds} milisaniye");
128     event1.Set();
129 }

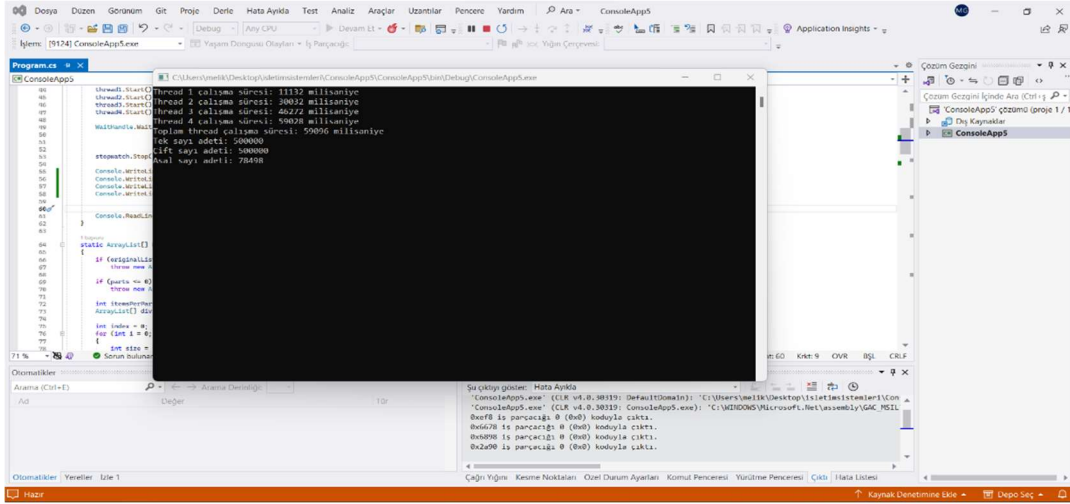
```

6.2. Aşama:

- Bu aşamada 85. satırda ilgili threadin süre sayacı başlatılıyor.
- Devamı ise asal sayı bulma şeklinde devam etmektedir.
- Bulunan asal sayılar `asalarray` listesine, tek sayılar `tekarray` listesinde, çift sayılar ise `çiftarray` listesine eklenmektedir.
- Burada eklenirken `lock` işlemi kullanılmaktadır yani ilgili listeyi sadece ilgili thread kullanabilsin diye.
- 127 ile 129. satırlar arasında ise ilgili thread için başlatılan süre sayacı durdurulup ölçüm yapılıyor ve `event1.Set()`; komutu ile de threadin sinyal durumu güncellenmektedir.

İşlemler neticesinde örnek konsol çıktısı şekildeki gibidir:

- Lock ifadesini kullandığımızda çıktılar şekildeki gibidir. Peki neyi farketmeliyiz süre çıktıları yanı sıra locksuz koddaki veri kaybı burada yok lock fonksiyonunun önemi.



2022 – 2023 Akademik Yılı Yazılım Mühendisliği İşletim Sistemleri Vize Ödevi

Soru 2

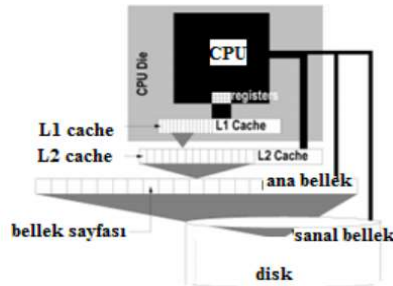
***Sorulan sorular farklı literatürlerde gerekli araştırmalar yapıp ilgili soru hakkında birden fazla literatürde bulunan cevaplar verilmiştir.**

1 - Sanal Bellek Kavramları:

-Sanal bellek nedir, nasıl çalışır, işletim sistemi açısından avantajları ve dezavantajları nelerdir? Bu konuda kapsamlı bir literatür taraması yaparak kavramları açıklayınız.

Google Akademik ile Literatür Taraması:

Sanal Bellek Nedir: Ana bellek üzerinde sabit sayıda tampondan oluşan bölgeye tampon havuzu denilmektedir (Sacco ve Schkolnik, 1982) [1]. Tampon havuzunda bulunan her tampon üzerinde bir disk bloğu bilgisi tutulmaktadır. Ayrıca kısıtlı olan ana bellek bölgesinin yetersiz kaldığı durumlarda disk üzerinde bir bölge ayrılarak ana bellek gibi kullanılabilir. Ayrılmış olan bu bölgeye de sanal bellek adı verilmektedir (Peter, 1970) [2].



Şekil 1. Bilgisayar üzerindeki donanımsal yapı (Stefan, 2002)

Şekil-1: [3]

Sanal Bellek Nasıl Çalışır: Sanal bellek, bilgisayarınızın sabit disk alanının bir bölümünü kullanarak RAM bellek alanını arttıran bir teknolojidir. Sanal bellek, bilgisayarınızın RAM bellek alanı dolu olduğunda, işletim sisteminin sabit diskinizdeki bir alanı geçici olarak RAM gibi kullanmasına izin verir.

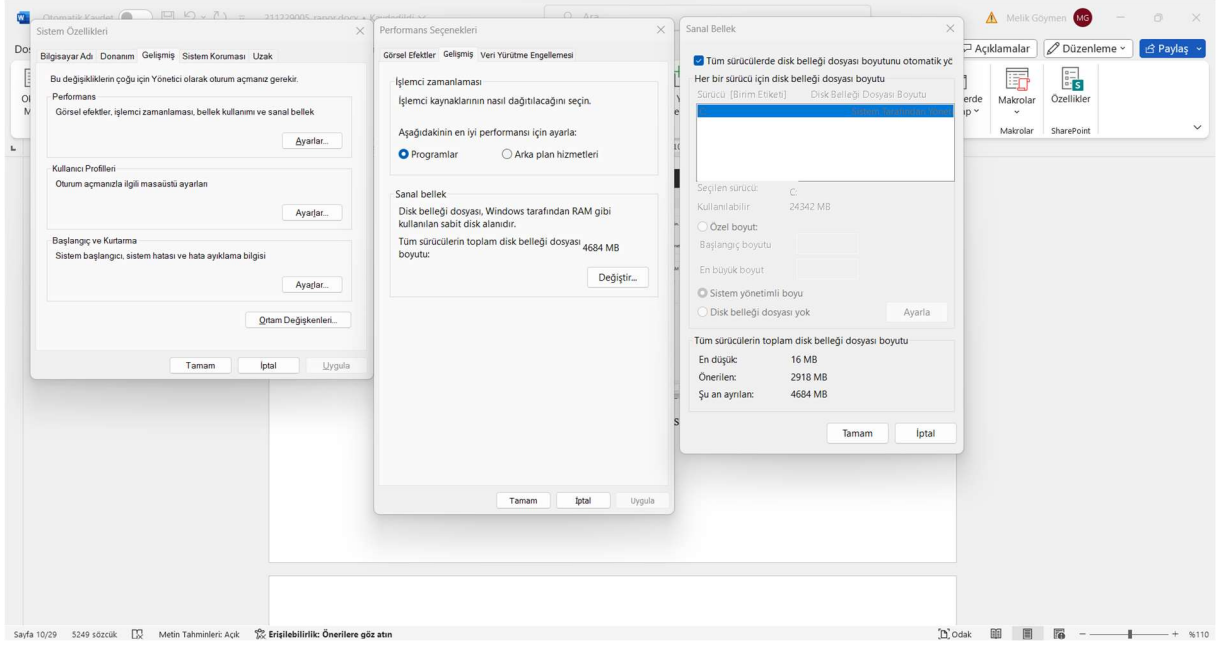
Sanal Bellek Avantajları: Sanallaştırma, bilişim kaynaklarının (işlemci, depolama, ağ, bellek, platform, sunucu, masaüstü, uygulama vb.) gerçekte var olan kaynağın değil; gerçek kaynağa dayandırılarak tanımlanmış olan soyut halinin, kullanıcılara sunulması olarak tanımlanabilir. Böylece gerçek kaynak, göreceli olarak daha az kapasiteli çok sayıda sanal kaynak olarak kullanılabilir [4]. Sunucuyu barındıracak ortamı veya donanımı sanallaştırmak olmak üzere iki tür temel sanallaştırma ortamı bulunmaktadır [5].

Sanal Bellek Dezavantajları: Sanal belleğin başlıca dezavantajları, performansla ilgilidir. Sanal bellek, fiziksel RAM yetersiz kaldığında devreye girer ve sabit diskin bir kısmını bellek olarak kullanır. Ancak, sabit diskler (özellikle HDD'ler), RAM'e göre çok daha yavaş erişim hızlarına sahiptirler. Bu durum, veri erişiminde önemli gecikmelere ve genel sistem performansında düşüşlere yol açabilir. Özellikle bellek yoğun uygulamalar kullanılırken, sistem hantallaşabilir ve tepki süreleri artabilir. Ayrıca, sabit diskin sürekli olarak sanal bellek olarak kullanılması, diskin ömrünü azaltabilir ve sistem stabilitesini olumsuz etkileyebilir. Kısacası, sanal bellek, acil bir çözüm olarak işlev görse de uzun vadede RAM kapasitesinin artırılmasına kıyasla daha az etkili bir yoldur.

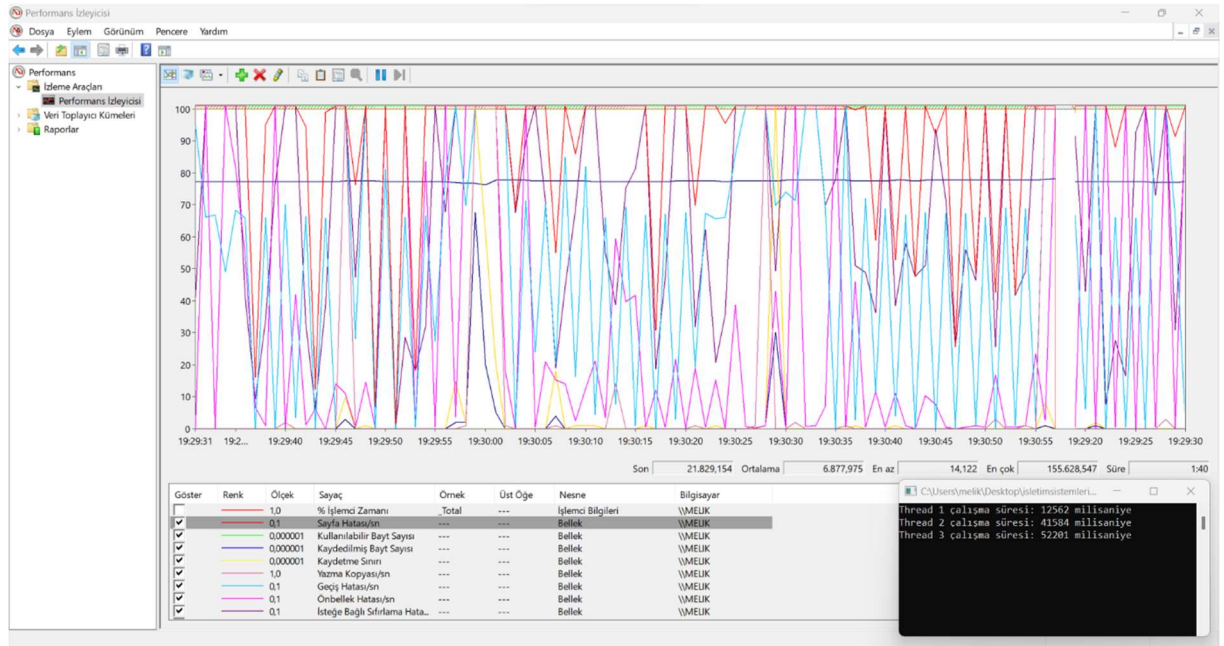
Kendi Edindiğim Öğrenimler: Windows işletim sistemi ile çalışan kişisel bilgisayarımda sanal bellek ayarları ile bazı değişiklikler yapmış bulunmaktayım örn:

Sanal Belleğin Ayarlarını Kontrol Etme:

Windows: Kontrol Paneli > Sistem > Gelişmiş sistem ayarları > Performans Ayarları > Gelişmiş > Sanal Bellek şeklinde ilgili pencereye yol aldığımızda açılan sayfa:



Şekil 2: Kişisel bilgisayarda sanal bellek ayarlarının gösterimi.



Şekil 3: Thread kodu çalıştığında bellek ve diğer değişkenlerin değer grafiği.

Çıkarımlar:

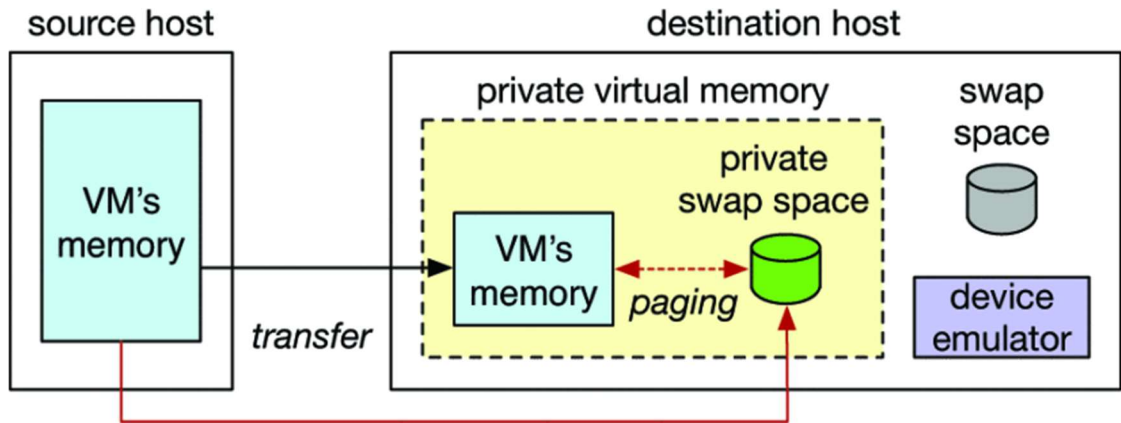
*Kişisel bilgisayarımnda sanal bellek yönetimi otomatik olarak gerçekleşmektedir.

*Anlık olarak: en düşük 16MB, önerilen 2918MB, şu an ayrılan 4684MB şeklinde ölçüldüğünü ve saptandığını fark ettim.

*Bilgisayarımın koşturduğum sisteme göre bellek değerlerin ve diğer bir çok performans değerlerini Performans İzleyicisi uygulamasından izleyebilirim.

IEEE (Institute of Electrical and Electronics Engineer) Xplore ile Literatür Taraması:

Sanal Bellek Nedir: Hedef ana makine yeterli belleğe sahip değilse, bellek-sanallaştıran VM göçü mümkündür. Bu, hedef ana makinede sanal belleği kullanır. VM'nin belleğinin bir kısmını ikincil depolamadaki takas alanında saklayabilir ve gerekli miktarda belleği kullanabilir. VM, takas alanındaki bellek verilerini gerektiğinde, bir sayfa içine alma işlemi gerçekleştirilir ve veriler fiziksel belleğe taşınır. Bunun karşılığında, fiziksel bellekteki muhtemelen erişilmeyecek veriler takas alanına taşınır. Ancak, geleneksel sanal belleğin VM göçü için uygun olmadığı belirtilmiştir [6].



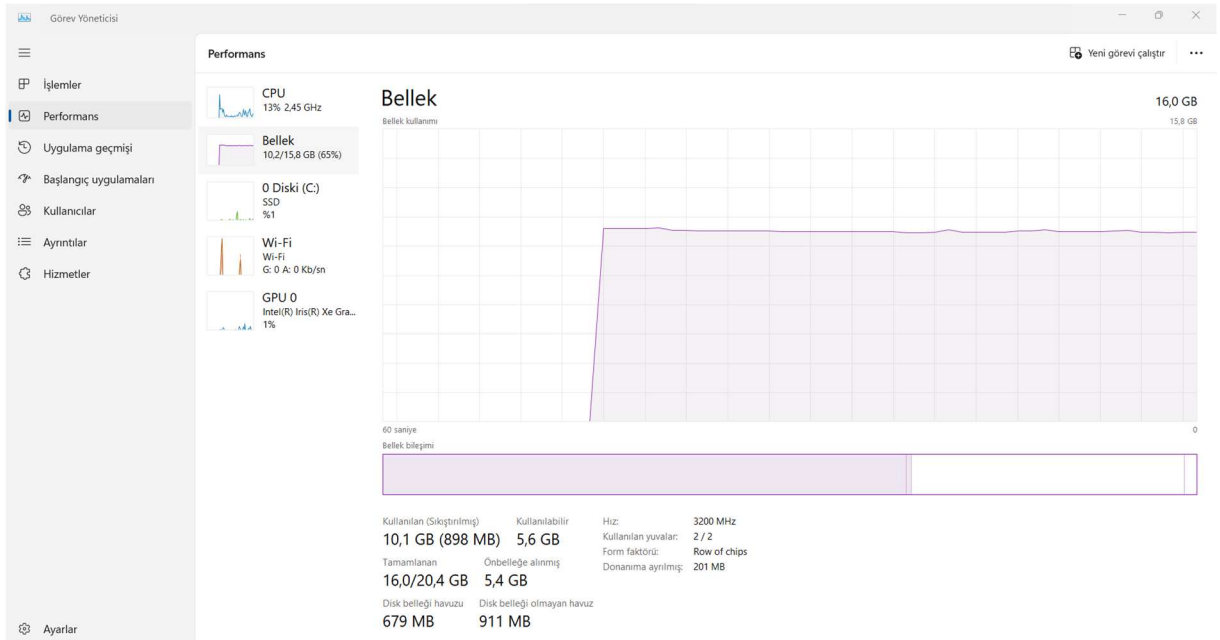
Şekil 4: Özel sanal bellek kullanarak belleği sanallaştıran VM geçişi.

Sanal Bellek Dezavantajları: Son zamanlarda, Altyapı olarak Hizmet (IaaS) bulutları, büyük miktarda belleğe sahip sanal makineler (VM'ler) sağlamaktadır. Örneğin, Amazon EC2 tarafından sağlanan VM'lerin belleği 24 TB'a kadar çıkabilmektedir. Bir VM çalıştıran bir ana makine bakım altına alındığında, VM'in başka bir ana makineye önceden taşınarak çalışmasının devam ettirilmesi mümkündür. VM göçü, bir VM'in durumunu, örneğin sanal CPU'ları, belleği ve sanal cihazları transfer eder ve VM'i hedef ana makinede yeniden başlatır. Bu nedenle, VM göçü, VM'in tüm belleğini barındırabilecek yeterli belleğe sahip büyük bir ana makine gerektirir. Ancak, ara sıra olan VM göçlerinin hedefi olarak bu tür büyük ana makineleri her zaman korumak

maliyet açısından verimli değildir. Özellikle, özel bulutlar, yeterli sayıda büyük ana makine hazırlayamayabilir [7].

Sanal Bellek Avantajları: Bu sorunları gidermek için bölünmüş göç [1] önerilmiştir. Bu yöntem, bir VM'nin belleğini böler ve bellek parçalarını daha küçük ana makineler arasında transfer eder. Muhtemelen erişilecek bellek verilerini ana ana makineye ve geri kalan bellek verilerini yardımcı ana makineler arasında aktarır. Eğer VM, göç sonrasında yardımcı ana makinelerdeki bellek verilerine ihtiyaç duyarsa, bu veriler uzaktan sayfalama yoluyla ana ana makineye aktarılır. VM göçü sırasında herhangi bir sayfalama olmadığı için, bölünmüş göç göç performansını artırabilir. Ayrıca, gerekli bellek verileri önceden ana ana makineye aktarıldığı için göç eden VM'nin çalışma performansını artırabilir [7].

Çıkarımlar:



Şekil 5: Kişisel bilgisayar bellek performansı.

*Kişisel bilgisayarımın bellek yönetimini Görev Yöneticisi sekmesinden izleyip çıkarımlarımı yaptım.

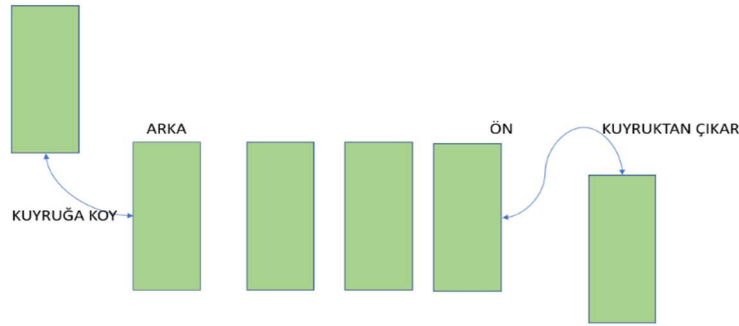
2- Sayfa Değiştirme Algoritmaları:

-FIFO, Optimal, LRU gibi sayfa değiştirme algoritmalarını araştırın. Her bir algoritmanın avantajlarını ve dezavantajlarını belirleyerek karşılaştırmalı bir analiz yapınız.

ArXiv, Google Akademi ve Academia.edu ile Literatür Taraması:

FIFO Sayfa Değiştirme Algoritması (FIFO Page Replacement Algorithm (İlk giren ilk çıkar sayfa değiştirme algoritması)): Sayfalama algoritmalarının düzgünlüğünü inceliyoruz. İstek sırasının bozulması nedeniyle sayfa hatalarının sayısı ne kadar artabilir? Sayfa hatalarındaki maksimum artış, istek dizisindeki değişikliklerin sayısıyla orantılıysa, sayfalama algoritmasına düzgün adını veriyoruz. Ayrıca bir algoritmanın düzgünlüğünü ölçen niceliksel düzgünlük kavramlarını da tanıtıyoruz. Deterministik ve rastgele talep çağrıları ve rekabetçi algoritmaların düzgünlüğüne ilişkin alt ve üst sınırları türetiyoruz. Son derece rekabetçi deterministik algoritmalar arasında LRU alt sınırla eşleşirken FIFO üst sınırla eşleşir. Partition, Equitable veya Mark gibi iyi bilinen rastgeleleştirilmiş algoritmaların düzgün olmadığı gösterilmiştir. Smoothed-LRU ve LRU-Random adı verilen iki yeni rastgeleleştirilmiş algoritma sunuyoruz. Düzleştirilmiş-LRU, ödünleşimin bir parametre tarafından kontrol edildiği akıcılık için rekabet gücünden fedakârlık edilmesine olanak sağlar. LRU-Random, daha sorunsuz olmasına rağmen en az herhangi bir deterministik algoritma kadar rekabetçidir [8]. Sayfalama, küçük ve hızlı bir bellek olan önbellek (cache) ve büyük ancak yavaş bir bellek olan ana bellekten oluşan iki seviyeli bir bellek sistemini modellemektedir. Bir programın çalışması sırasında, veriler sayfalar olarak bilinen veri birimleri halinde önbellek ile bellek arasında transfer edilir. Önbelleğin boyutu, sayfa olarak k ifade edilir ve belleğin boyutu sonsuz kabul edilebilir. Sayfalama problemine giriş, önbellekte bulunması gereken bir dizi sayfa talebidir. Bir sayfa talebi geldiğinde ve bu sayfa zaten önbellekteyse, herhangi bir işlem gerekmez. Bu durum "isabet" olarak adlandırılır. Aksi takdirde, sayfa bellekten önbelleğe getirilmelidir, bu da önbellekten başka bir sayfanın çıkarılmasını gerektirebilir. Bu, "sayfa hatası" veya "kaçırma" olarak bilinir. Sayfalama algoritması, hata sayısını en aza indirmek için önbellekte hangi sayfaların tutulacağına karar vermelidir. Bir sayfalama algoritması,

yalnızca tam dolu bir önbellekte hata olduğunda bir sayfayı önbellekten çıkaran bir sayfa hatası durumunda talep sayfalama olarak adlandırılır. Talep dışı herhangi bir sayfalama algoritması, performansı feda etmeden talep sayfalama yapılabilir duruma getirilebilir [9]. Bu algoritmaya göre bir sayfa ihlali (page fault) olduğunda, yani hafızada (RAM) bulunmayan bir sayfaya erişilmek istendiğinde, yani diskteki bir sayfaya erişilmek istendiğinde, Diskten ilgili sayfa hafızaya (RAM) yüklenirken, hafızadaki en eski sayfa yerine yüklenir ve bu en eski sayfa da diske geri yazılır [11].



Şekil 6: FIFO Sayfa değiştirme algoritması anlatım şekli.

FIFO (First-In, First-Out) Avantajları:

- 1. Basitlik:** FIFO, uygulanması ve anlaşılması kolay bir algoritmadır.
- 2. Adil Kullanım:** Tüm sayfalar eşit süre boyunca bellekte tutulur, bu da adil bir kullanım sağlar.
- 3. Öngörülebilir Davranış:** Sayfaların bellekten ne zaman çıkarılacağı öngörülebilir.

FIFO (First-In, First-Out) Dezavantajları:

- 1. Sık Kullanılan Sayfaların Atılması:** Sık kullanılan veya önemli sayfalar, bellekte yeterince uzun süre kalmadan atılabilir.
- 2. Performans Sorunları:** FIFO, sayfa hata oranını azaltmada etkili değildir, bu da performans sorunlarına yol açabilir.

3. Kötü Hafıza Kullanımı: Bellekteki verimli kullanımı sağlamada zayıf olabilir, özellikle değişen veya yoğun erişim desenlerinde.

LRU Sayfa Değiştirme Algoritması (Least Recently Used (LRU) Page replacement (En nadir kullanılan sayfa değiştirme algoritması)): Bu makale, LRU-K yöntemi olarak adlandırılan, veritabanı disk arabelleğe alma işlemine yeni bir yaklaşım sunmaktadır. *LRU-K'nin temel fikri, popüler veritabanı sayfalarına yapılan son K referanslarının zamanlarını takip etmek ve bu bilgiyi kullanarak referansların varışlar arası zamanlarını sayfa sayfa istatistiksel olarak tahmin etmektir.* LRU-K yaklaşımı, nispeten standart varsayımlar altında optimal istatistiksel çıkarımı gerçekleştirmesine rağmen oldukça basittir ve çok az muhasebe yükü gerektirir. Simülasyon deneyleriyle gösterdiğimiz gibi, LRU-K algoritması, sık ve seyrek başvurulmuş sayfalar arasında ayırma yapma konusunda geleneksel ara belleğe alma algoritmalarını geride bırakıyor. Aslında LRU-K, bilinen erişim frekanslarına sahip sayfa kümelerinin özel olarak ayarlanmış boyutlardaki farklı arabellek havuzlarına manuel olarak atandığı ara belleğe alma algoritmalarının davranışına yaklaşılabılır. Bununla birlikte, bu tür özelleştirilmiş tamponlama algoritmalarının aksine, LRU-K yöntemi kendi kendini ayarlar ve iş yükü özelliklerine ilişkin harici ipuçlarına dayanmaz. Ayrıca LRU-K algoritması, değişen erişim modellerine gerçek zamanlı olarak uyum sağlar [10].

Veritabanı sistemleri, diskten okunduktan ve belirli bir uygulama tarafından erişildikten sonra, disk sayfalarını bir süre bellek tamponlarında tutar. Bunun amacı, popüler sayfaları bellekte tutarak disk giriş/çıkışını azaltmaktır. Gray ve Putzolu'nun "Beş Dakika Kuralı"nda şu ticaret-off dile getirilmiştir: Bellek tamponları için daha fazla ödemeye, sistem için disk kollarının maliyetini azaltmak adına belirli bir noktaya kadar razıyızdır. Kritik tamponlama kararı, diskten okunmak üzere yeni bir tampon yuvasına ihtiyaç duyulduğunda ve tüm mevcut tamponlar kullanımda olduğunda ortaya çıkar: Hangi mevcut sayfa tampondan çıkarılmalıdır? Bu, sayfa değiştirme politikası olarak bilinir ve farklı tamponlama algoritmaları, uyguladıkları değiştirme politikasının türüne göre adlandırılır. Çoğu ticari sistem tarafından kullanılan algoritma, En Az Son Kullanılan (LRU) olarak bilinir. Yeni bir tampona ihtiyaç duyulduğunda, LRU politikası en uzun süredir erişilmemiş sayfayı tampondan çıkarır. LRU tamponlaması, başlangıçta talimat mantığındaki kullanım kalıpları için geliştirilmiş olup, veritabanı ortamına her zaman iyi uyum sağlamaz. Gerçekten de LRU tamponlama algoritmasının, son referans

zamanına dayanarak hangi sayfanın tampondan çıkarılacağına karar verme konusunda yetersiz bilgiye sahip olması gibi bir problemi vardır. Özellikle, LRU, nispeten sık referanslara sahip sayfalar ile çok nadir referanslara sahip sayfalar arasındaki farkı, sistem nadiren referans verilen sayfaları uzun bir süre için tamponda tuttuğu noktaya kadar ayırt edemez [10]. Bu algoritmaya göre bir sayfa ihlali (page fault) olduğunda, yani hafızada (RAM) bulunmayan bir sayfaya erişilmek istendiğinde, yani diskteki bir sayfaya erişilmek istendiğinde, Diskten ilgili sayfa hafızaya (RAM) yüklenirken, hafızadaki en az erişilen sayfa yerine yüklenir ve bu en az kullanılan sayfa da diske geri yazılır [11].

LRU için

String	3	4	5	1	2	3	4	5	3	4	5	5
F1	3	3	3	1	1	1	4	4	4	4	4	4
F2		4	4	4	2	2	2	5	5	5	5	5
F3			5	5	5	3	3	3	3	3	3	3
Hata	VAR	VAR	VAR	VAR	VAR	VAR	VAR	VAR	YOK	YOK	YOK	YOK

* Sayfa hatası değeri: 8 dir. Sayfa hatası oranı ise; $8 / 12 = 0,66$ dir.

Şekil 7: LRU sayfa değiştirme algoritması anlatım şekli.

LRU (Least Recently Used) Avantajları:

- 1. Verimlilik:** LRU, sık kullanılan sayfaları bellekte tutarak verimliliği artırır.
- 2. Azaltılmış Sayfa Hataları:** Daha az sayfa hatası oluşur, çünkü sık erişilen sayfalar genellikle bellekte kalır.
- 3. Adaptif Davranış:** Çalışma zamanında erişim desenlerine adapte olabilme yeteneği sağlar.

LRU (Least Recently Used) Dezavantajları:

- 1. Yönetim Maliyeti:** LRU'nun takip etmesi gereken ek bellek ve işlem maliyetleri vardır.
- 2. Uygulama Karmaşıklığı:** Algoritmanın uygulanması, basit algoritmalara göre daha karmaşıktır.

3. Zayıf Yaşlanma Politikası: Uzun süre bellekte kalan, ancak artık sık kullanılmayan sayfaların gereksiz yere tutulması problemi olabilir.

Optimal Sayfa Değiştirme Algoritması (Optimal Replacement (Mükemmel Sayfa Değiştirme Algoritması)): Bu algoritma, hiçbir zaman gerçekleştirilemeyecek hayali bir algoritmadır. Akademik olarak ortaya atılmıştır ve algoritmanın çalışması için daha sonra gelecek olan sayfa ihtiyaçlarının önceden bilinmesi gerekir. Bu sayfa değiştirme algoritmasına göre bir sayfa ihlali (page fault) olduğunda, yani hafızada (RAM) bulunmayan bir sayfaya erişilmek istendiğinde, yani diskteki bir sayfaya erişilmek istendiğinde, Diskten ilgili sayfa hafızaya (RAM) yüklenirken, hafızada bundan sonra

En Uygun Algoritması (Optimal page replacement)

- Bu algoritmada, gelecekte istenilebilecek en son sayfa seçilir ve sanal belleğe taşınır.
- Toplam 6 tane sayfa hatası oluştu.

Ana belleğe taşınacak sayfalar	2	3	2	1	5	2	4	5	3	2	5	2
Çerçeve 1	2	2	2	2	2	2	4	4	4	2	2	2
Çerçeve 2		3	3	3	3	3	3	3	3	3	3	3
Çerçeve 3				1	5	5	5	5	5	5	5	5
Sayfa hatası	X	X		X	X		X			X		

en uzun süre erişilmeyecek olan yerine yüklenir ve bu en az kullanılan sayfa da diske geri yazılır [11].

Şekil 8: Optimal sayfa değiştirme algoritması anlatım şekli.

Optimal Avantajları:

- 1. En Az Sayfa Hatası:** Optimal algoritma, teorik olarak mümkün olan en az sayfa hatasını sağlar.
- 2. İdeal Performans Referansı:** Diğer algoritmaların performansını değerlendirmek için bir referans noktası sağlar.
- 3. Etkin Bellek Kullanımı:** Bellek kaynaklarını en verimli şekilde kullanır.

Optimal Dezavantajları:

- 1. Gerçek Zamanlı Uygulanabilirlik:** Gelecekteki sayfa isteklerini önceden bilmek gerektiği için pratikte uygulanamaz.

2. Tahmin Edilemezlik: Gelecekteki erişimleri tahmin etmek zor olduğundan, gerçek dünya senaryolarında kullanılamaz.

3. Uygulama Karmaşıklığı: Teorik olarak ideal olsa da gerçek uygulamalar için aşırı karmaşıktır.

FIFO (First-In, First-Out), LRU (Least Recently Used) ve Optimal sayfa değiştirme algoritmalarını kendi aralarında Performans ve Verimlilik, Uygulanabilir ve Karmaşıklık, Uygunluk perspektiflerinden incelediğim elde ettiğim sonuçlar ve kazanımlar şunlardır:

1. Performans ve Verimlilik: Optimal algoritma teorik olarak en iyi performansı sunmaktadır, çünkü gelecekteki sayfa isteklerini biliyor ve en az sayfa hatası üreterek bu işlemi gerçekleştiriyor. LRU, gerçek zamanlı senaryolarda daha iyi performans göstermektedir çünkü en son kullanılan sayfaları bellekte tutarak bu işlemleri gerçekleştirir. FIFO ise daha düşük performans göstermektedir, çünkü sadece giriş sırasına göre işlem yapmaktadır.

2. Uygulanabilirlik ve Karmaşıklık: FIFO en basit uygulamaya yapısına sahiptir, ancak bu basitlik, etkin bellek kullanımı ve düşük sayfa hata oranı açısından dezavantajları ortaya çıkartmaktadır. LRU, FIFO'ya göre daha karmaşık bir uygulamaya sahip olduğu halde, daha iyi bir performans sunar. Bunun nedeni ise; LRU'nun sayfaların kullanım sıklığını ve zamanını dikkate almasıdır. FIFO yalnızca sayfaların belleğe giriş sırasına göre işlem yaparken, LRU en son ne zaman kullanıldıklarına bakarak sayfaları yönetmektedir. Bu yaklaşım, sık kullanılan ve önemli sayfaların bellekte kalmasını sağlayarak sayfa hatalarını azaltır ve bellek erişimini optimize etmesini sağlamaktadır. Optimal algoritma ise pratikte uygulanabilir değildir, çünkü gelecekteki sayfa isteklerini bilmek gereklidir.

3. Uygunluk: FIFO, öngörülebilir ve adil bir bellek yönetimi sunar, ancak sık kullanılan sayfaları atabilir. FIFO algoritması, sayfaları belleğe giriş sırasına göre yönetmektedir. Bu, ilk giren sayfanın ilk çıkarılacağı anlamına gelir. Bu yöntem öngörülebilir ve adildir çünkü her sayfa aynı süre boyunca bellekte tutulmaktadır. Ancak, bu yaklaşım, bir sayfanın ne kadar sık kullanıldığını veya ne kadar önemli olduğunu dikkate almaz. Bu nedenle, FIFO, sık kullanılan veya kritik öneme sahip sayfaları, bellekte daha az süre

kalmış olsalar bile atabilir. Bu, özellikle yoğun bellek erişim desenlerinde verimsizliklere yol açabilir. LRU, bellek yönetimini kullanım sıklığına göre optimize eder yani sayfaların bellekte ne kadar süre tutulacağını belirlemek için son kullanım zamanlarını kullanarak optimize işlemlerini gerçekleştirir. Optimal ise teorik olarak en etkin bellek yönetimini sunar, ancak bu, gerçek dünyada uygulanabilir değildir. Bunun sebebi ise, Optimal sayfa değiştirme algoritması, gelecekteki sayfa isteklerini önceden bilebilmeyi gerektirir. Gerçek dünyada bu, mümkün değildir çünkü gelecekte hangi sayfaların isteneceğini önceden tahmin etmek pratikte uygulanabilir bir yöntem değildir. Bu nedenle, Optimal algoritma daha çok teorik bir model olarak kullanılır ve gerçek zamanlı sistemlerde uygulanabilir bir yöntem değildir. Bu algoritma, ideal koşullarda mümkün olan en düşük sayfa hata oranını sağlamak için tasarlanmış bir kavramsal modeldir.

Kullanım Alanları Örnekleri:

FIFO: Az değişkenlik gösteren veri erişim desenleri olan uygulamalarda kullanışlıdır.

1. Basit Web Siteleri: Statik içerikli web siteleri, ziyaretçi trafiği az ve sabit olduğunda FIFO kullanımını için uygun olabilir.

2. Arşivleme Sistemleri: Düzenli aralıklarla erişilen, ancak sıklıkla güncellenmeyen arşiv dosyalarını yöneten sistemler.

3. Belirli Endüstriyel Kontrol Sistemleri: Sabit ve tahmin edilebilir veri toplama ve işleme desenlerine sahip sistemler.

LRU: Veri erişim desenlerinin sık değiştiği, örneğin web sunucuları veya çok kullanıcıli veri tabanı sistemleri gibi ortamlarda tercih edilir.

1. Çok Kullanıcıli Veri Tabanı Sistemleri: Sürekli olarak farklı veri tabanı sorguları alan ve çeşitli kullanıcı taleplerine cevap veren sistemler.

2. Web Sunucuları: Farklı içeriklere sürekli erişim sağlanan ve sıkça güncellenen web sitelerini barındıran sunucular.

3. Uygulama Sunucuları: Çok sayıda kullanıcı tarafından sürekli erişilen ve dinamik içerik sunan uygulama sunucuları.

Optimal: Teorik olarak en iyi performansı sunar, ancak gelecekteki erişimleri önceden bilmek gerektiğinden gerçek uygulamalarda kullanılamaz. Araştırma ve teorik analizlerde, diğer algoritmaların performansını değerlendirmek için ideal bir referanstır.

1. Akademik ve Araştırma Çalışmaları: Bilgisayar bilimi ve veri tabanı sistemleri üzerine araştırma yapılırken, diğer algoritmaların performansını karşılaştırmak için bir referans noktası olarak kullanılır.

2. Algoritma Simülasyonları: Bilgisayar mimarisi veya işletim sistemleri derslerinde, öğrencilere sayfa değiştirme stratejilerinin etkinliğini öğretmek için simülasyonlar kullanılır.

3. Performans Testleri: Yeni geliştirilen sayfa değiştirme algoritmalarının teorik performans sınırlarını belirlemek için kullanılır.

3 – Sanal Bellek Hiyerarşisi:

-Bellek hiyerarşisinin nasıl işlediğini ve ana bellek (RAM), ikincil bellek (disk), sayfa tablosu gibi kavramların birbirleriyle nasıl etkileşimde bulunduğunu açıklayınız.

Öncelikle ilgili soruda geçen kavramlar ve bilgisayar donanımları nedir:

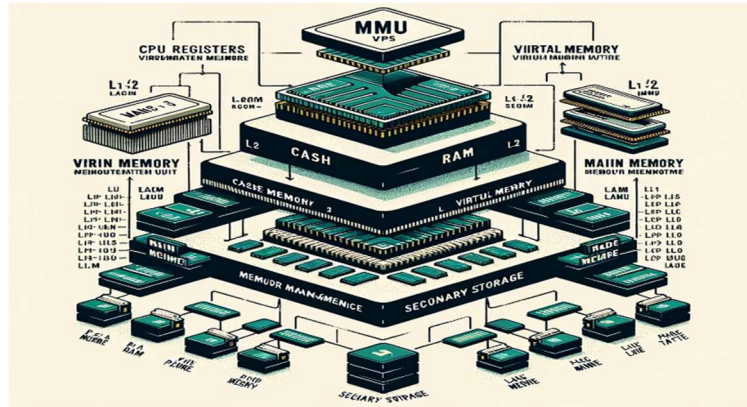
RAM (Random Acces Memory) Rastgele Erişimli Bellek: RAM bilgilerin geçici olarak depolandığı bir hafıza türüdür. Bilgisayarlar genellikle o an üzerinde çalıştıkları programlar ve işlemlerle ilgili bilgileri RAM denen bu hafıza parçasında tutarlar. RAM ve sabit sürücü temel olarak aynı bilgileri saklarlar, ancak işlemcinin RAM'deki bilgilere erişme ve onları işleme hızı, sabit sürücüdeki bilgilere erişme ve onları işleme hızından çok daha büyüktür.

İkincil Bellek (Disk): Bilgisayar depolama hiyerarşisinde ikincil depolama sistemi olarak işlev gören, verileri kalıcı olarak depolamak için tümleşik devre tertibatlarını kullanan bir veri depolama aygıtıdır [12].

Sayfa Tablosu: Sayfa tablosu, sanal bellek yönetiminde kritik bir rol oynayan bir veri yapısıdır. İşletim sistemi tarafından kullanılır ve sanal adresleri fiziksel adreslere çevirmek için kullanılır. Bir program çalıştığında, kod ve veriler sanal adreslerle ifade edilir. Sayfa tablosu, bu sanal adreslerin hangi fiziksel RAM adreslerine karşılık geldiğini belirler. Böylece, programların daha büyük ve sürekli bir bellek alanında çalışıyormuş gibi hissetmelerini sağlar. Sayfa tablosu, RAM'de olmayan verilerin diskten yüklenmesi gerektiğinde (sayfa hatası) de devreye girer.

Donanımlar ve kavramlar arasındaki ilişki ve çalışma prensibi nedir: Sanal bellek hiyerarşisi, farklı bellek türlerinin ve hızlarının bir araya gelmesiyle oluşan bir sistemdir. Bu hiyerarşi, işlemcilerin hızı ve işlem gücü arttıkça daha da önem kazanır. Bellek sınırlı bir kaynak olduğu için, daha fazla uygulamanın kullanılmasıyla performans düşebilir veya durma noktasına gelebilir. İşte burada sanal bellek devreye girer ve sistemdeki RAM'i taklit ederek sabit sürücünün bir bölümünü kullanır. Sanal bellek, sabit disk sürücüsünü geçici depolama alanı gibi kullanır ve böylece yazılımın ek bellek olarak kullanılmasına olanak tanır. Sanal bellek hiyerarşisinde farklı bellek türleri vardır. Bu bellek türleri, işlemciye yakınlık ve hız, düşük gecikme ve maliyet bakımından yukarıdan aşağıya doğru sıralanabilir. Örneğin, CPU içindeki yazmaçlar, L1 ve L2 önbellekler, ana bellek (DRAM), yerel ve uzaktan erişimli ikincil depolama birimleri hiyerarşinin farklı seviyelerini oluşturur. Her bir bellek türü, bir üst seviyedeki bellek türünü önbellek olarak kullanır. Örneğin, yerel diskte kayıtlı bir dosya işlenirken ana bellek, işletim sistemi tarafından geçici depolama birimi olarak kullanılabilir. Bu hiyerarşinin en üst kısmında, hız açısından işlemciye en yakın olan bellek birimleri yer alır. L1 önbellek doğrudan işlemcinin içinde bulunurken, L2 önbellek genellikle işlemci kartına yerleştirilir ve bu nedenle erişim hızı L1'e göre biraz daha düşük, kapasitesi ise daha yüksektir. Hiyerarşinin üst kısımlarındaki bellek birimleri genellikle daha hızlı fakat küçük kapasitelidir, alt kısımlarındakiler ise daha büyük kapasiteli fakat hızları düşüktür. İşlemci bir veriyi ana bellekten okumak veya yazmak istediğinde, önce ön belleğe bakar. Eğer istenen veri varsa ve içeriği değiştirilmemişse bu veriyi okur. Yazma

sürecinde ise, önce ön bellekteki, sonra ise uygun bir zaman bulunduğunda ana bellekteki bir adresin içerikleri değiştirilir. Bu şekilde, bellek erişim hızı ve kapasitelerine göre farklılık gösteren bir hiyerarşi oluşturulmuş olur.



Şekil 9: Sanal bellek hiyerarşisini anlatan şekil.

4 – Sayfa Tablosu Analizi:

-Sayfa tablosu nedir, nasıl çalışır? Sayfa tablosu oluşturulurken kullanılan teknikleri inceleyip avantaj ve dezavantajlarını belirtiniz.

Sayfa Tablosu Nedir:

Sayfa tablosu, sanal bellek kullanan bilgisayar sistemlerindeki bir veri yapısıdır. İşletim sistemi, sanal adresleri fiziksel adreslere dönüştürmek için sayfa tablosunu kullanır. Bu, özellikle sanal adres alanının fiziksel bellekten daha büyük olduğu durumlarda önemlidir.

Sayfa Tablosu Nasıl Çalışır:

- 1. Adres Dönüşümü:** Sayfa tablosu, sanal bellek adreslerini (bir program tarafından kullanılan adresler) fiziksel bellek adreslerine (RAM'daki gerçek adresler) dönüştürür.
- 2. Sayfa Tablosu Girdileri:** Her tablo girdisi, sanal sayfa numarasını (VPN) ve eşleşen fiziksel sayfa numarasını (PPN) içerir. Ayrıca, erişim hakları, değiştirilme durumu ve sayfanın bellekte olup olmadığı gibi bilgileri de içerebilir.
- 3. Arama İşlemi:** Bir program belirli bir sanal adresi talep ettiğinde, işletim sistemi bu adresi sayfa tablosunda arar ve eşleşen fiziksel adresi bulur. Eğer

sayfa bellekte yoksa, sayfa hatası (page fault) oluşur ve ilgili sayfa diske kaydedilir veya diskten yüklenir.

Sayfa Tablosu Oluşturma Teknikleri:

- 1. Basit Sayfa Tablosu:** Tek bir global tablo tüm sanal sayfaları fiziksel sayfalara eşler. Bu yöntem basit ve anlaşılırdır, ancak büyük bellek alanları için verimsiz olabilir.
- 2. Çok Düzeyli Sayfa Tablosu:** Büyük adres uzaylarını yönetmek için birden fazla sayfa tablosu kullanılır. Bu, sayfa tablolarının boyutunu azaltır ve bellek kullanımını optimize eder.
- 3. Tersine Çevrilmiş Sayfa Tablosu:** Her fiziksel sayfa için yalnızca bir girdi içerir. Bu, sayfa tablosunun boyutunu fiziksel bellek boyutuna orantılar, büyük sanal adres uzayları için daha verimlidir.
- 4. Hashed Sayfa Tablosu:** Büyük sanal adres uzaylarını daha verimli bir şekilde yönetmek için hash tablolarını kullanır.

Avantajlar ve Dezavantajlar

1. Basit Sayfa Tablosu:

Avantaj: Yapılandırması ve anlaşılması kolay.

Dezavantaj: Büyük sanal adres uzayları için verimsiz ve bellek tüketimi yüksek.

2. Çok Düzeyli Sayfa Tablosu:

Avantaj: Büyük adres uzaylarını daha etkili yönetir, bellek kullanımını azaltır.

Dezavantaj: Adres dönüşümü daha karmaşık hale gelir.

3. Tersine Çevrilmiş Sayfa Tablosu:

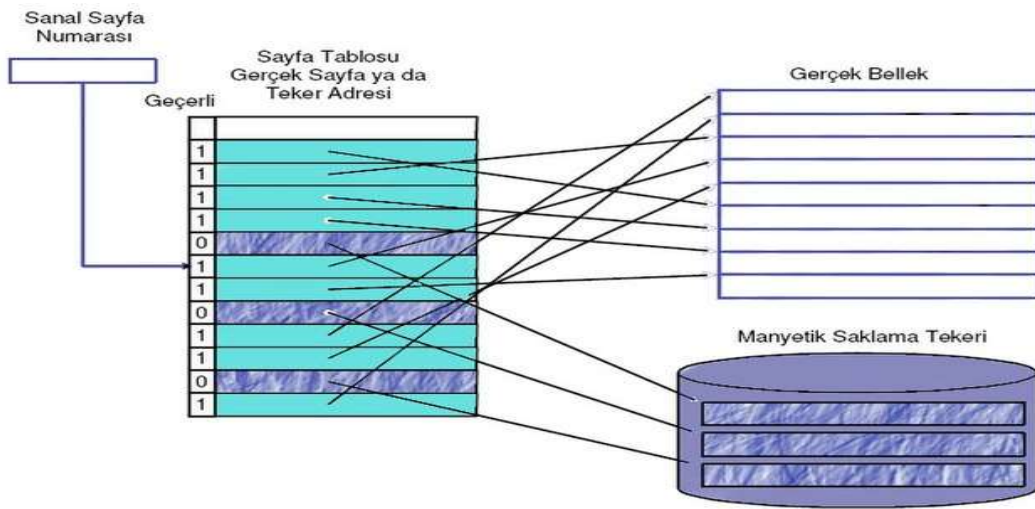
Avantaj: Bellek tüketimi düşük, büyük bellek alanlarını etkili bir şekilde yönetir.

Dezavantaj: Adres dönüşüm süreci daha karmaşık ve zaman alıcı olabilir.

4. Hashed Sayfa Tablosu:

Avantaj: Büyük sanal adres uzaylarını etkin şekilde yönetir, hızlı arama yapısı sağlar.

Dezavantaj: Hash çakışmaları yönetimi gerektirir ve karmaşık olabilir.



Şekil 10: Sayfa tablosu analizi örnek gösterim şekli.

Sayfa tablolarının seçimi, bellek büyüklüğü, adres uzayının boyutu ve sistem performans gereksinimlerine bağlı olarak değişir. Her teknik, belirli senaryolara ve sistem mimarilerine göre avantajlar ve dezavantajlar sunar.

5 – Bellek Fragmantasyonu:

-Bellek fragmantasyonu kavramını açıklayınız. İç ve dış fragmantasyon nedir? Sanal bellek yönetiminin bellek fragmantasyonuyla başa çıkma yöntemlerini araştırınız.

Bellek Fragmantasyonu Nedir:

Bellek fragmantasyonu, bir bilgisayar sisteminin belleğinin kullanımı sırasında meydana gelen ve belleğin etkin kullanımını engelleyen bir sorundur. Bu, belleğin parçalı ve kullanılamaz hale gelmesine yol açar.

İç ve Dış Fragmantasyon:

1. İç Fragmantasyon (Internal Fragmentation):

1. Bellek, bir sürecin ihtiyacından daha büyük bloklara bölündüğünde meydana gelir.
2. Her blokta, atanan belleğin kullanılmayan bir kısmı vardır.
3. Örneğin, bir süreç 5KB'lık bir bellek talep eder ancak en küçük kullanılabilir blok 10KB'dır. Geriye kalan 5KB kullanılmaz ve iç fragmantasyona yol açar.

2. Dış Fragmantasyon (External Fragmentation):

1. Toplamda yeterli boş bellek olsa bile, süreçlerin ihtiyaçları için yeterli ardışık boş alanın olmaması durumudur.
2. Zamanla, sürekli olarak belleğin ayrılması ve serbest bırakılması, küçük, kullanılamaz boşlukların oluşmasına neden olur.
3. Örnek: Bellekte 10KB boş yer var, ancak bu, 2KB'lık beş ayrı parçada mevcuttur. 6KB'lık bir sürecin talebi karşılanamaz.

Sanal Bellek Yönetiminin Bellek Fragmantasyonu ile Başa Çıkma Yöntemleri:

1. Sayfalama (Pagination):

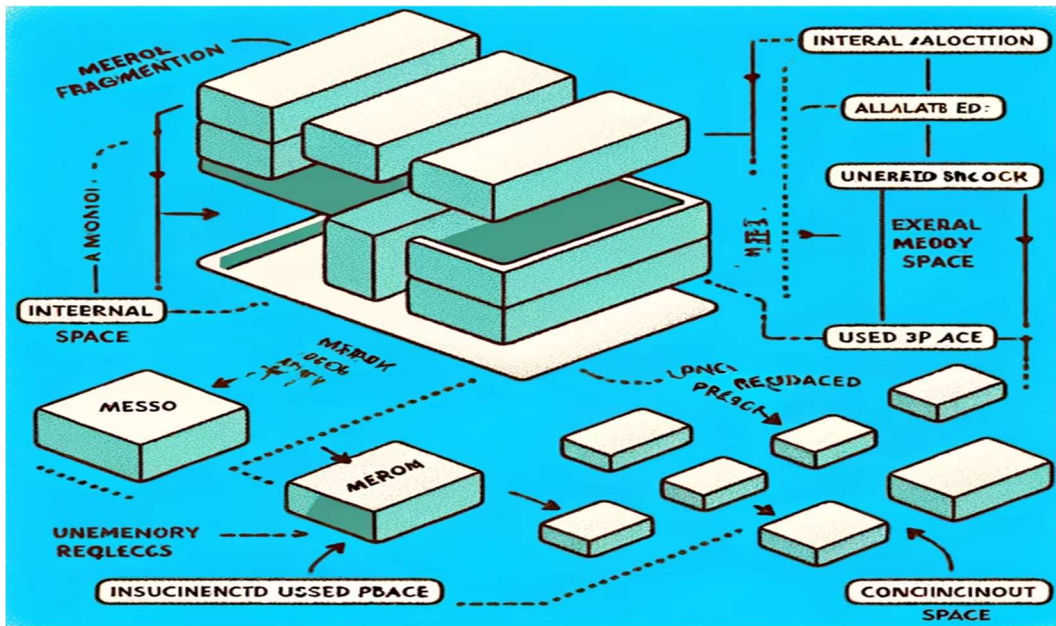
1. Bellek, sabit boyutlu bloklar (sayfalar) halinde bölünür.
2. Her süreç kendi sanal adres alanına sahiptir ve bu alan fiziksel belleğe sayfa sayfa eşlenir.
3. Sayfalama, dış fragmantasyonu ortadan kaldırır çünkü her sayfa, belleğin herhangi bir yerinde yer alabilir.

2. Segmentasyon (Segmentation):

1. Bellek, deęişken boyutlu segmentlere bölünür.
2. Her segment, bir sürecin mantıksal birimini (örneğin kod, veri, yığın) temsil eder.
3. Segmentasyon, iç fragmentasyonu azaltmaya yardımcı olur, çünkü her segment sadece gereken kadar bellek alır.

3. Sanal Bellek Kullanımı (Virtual Memory Usage):

1. Fiziksel bellekten daha büyük bir adres alanı sağlar.
2. Belleğin daha verimli kullanılmasına izin verir ve süreçlerin bellek ihtiyaçlarını daha iyi karşılar.
3. Aktif olmayan sayfalar disk gibi ikincil depolama alanına taşınarak, bellekte yer açılır ve fragmentasyon azaltılır.



Şekil 11: Bellek fragmentasyonunu anlatan bir görsel.

Bellek fragmentasyonu, belleğin etkin kullanımını engelleyen bir sorun olsa da modern işletim sistemlerinin bellek yönetimi teknikleri bu sorunu büyük ölçüde azaltmaktadır.

6 – Sanal Bellek Yönetimi Uygulama:

- İstedığınız bir işletim sistemi üzerinde belirli bir uygulamanın sanal bellek yönetimini inceleyiniz. Uygulama sırasında ortaya çıkabilecek durumları analiz ediniz.

Windows İşletim Sistemi Üzerinde Sanal Bellek Yönetimi:

Windows, sanal bellek yönetimini, her uygulamanın kendi adres alanına sahip olduğu ve bu alanın fiziksel belleğe sayfa sayfa eşlendiği bir sayfalama mekanizması üzerinden gerçekleştirir.

1. Sanal Bellek Yönetimi Süreci:

1. Sayfa Dosyası Oluşturma: Windows, sabit disk üzerinde bir sayfa dosyası (pagefile.sys) oluşturur. Bu dosya, fiziksel bellekten (RAM) taşan veriler için kullanılır.

2. Sayfa Hataları ve Taşıma: Eğer bir uygulama, bellekte olmayan bir sayfaya erişmeye çalışırsa, bir sayfa hatası meydana gelir. Bu durumda, işletim sistemi gerekli sayfayı sayfa dosyasından (veya uygulamanın executable dosyasından) belleğe taşır.

3. Bellek Yönetimi ve Sayfa Değiştirme: Windows, aktif olarak kullanılmayan sayfaları sayfa dosyasına taşıyarak, RAM'de yer açar. Bu, belleği daha etkin kullanmaya olanak tanır.

2. Uygulama Sırasında Ortaya Çıkabilecek Durumlar:

1. Yetersiz Sanal Bellek (Insufficient Virtual Memory): Eğer sayfa dosyası ve fiziksel bellek yetersizse, sistem performansı düşebilir. Bu durumda, kullanıcıdan sayfa dosyasının boyutunu artırması istenebilir.

2. Aşırı Sayfa Değiştirme (Thrashing): Eğer sistem sürekli olarak sayfaları bellekten diske ve diskten belleğe taşıyorsa, bu aşırı sayfa değiştirme (thrashing) olarak adlandırılır ve sistem performansını ciddi şekilde etkileyebilir.

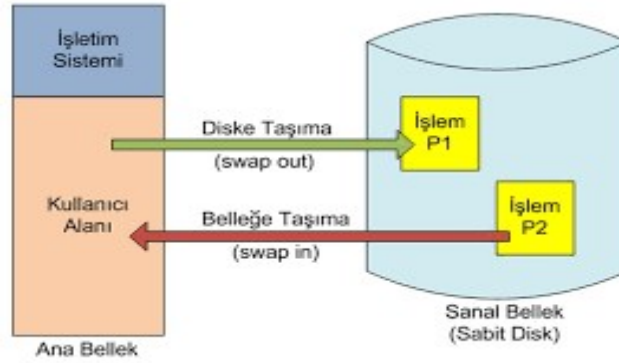
3. Bellek Sızıntıları (Memory Leaks): Uygulamalar hatalı bellek yönetimi nedeniyle gereksiz bellek alanı kaplayabilir. Bu, zamanla belleğin dolmasına ve sistem performansının düşmesine neden olabilir.

3. Önlemler ve Yönetim:

1. Sayfa Dosyası Boyutunu Yönetme: Kullanıcılar, sayfa dosyasının boyutunu manuel olarak ayarlayabilir veya sistemin otomatik yönetimine bırakabilir.

2. Uygulama Güncellemeleri: Bellek sızıntıları genellikle uygulama güncellemeleriyle düzeltilir. Kullanıcıların uygulamalarını düzenli olarak güncellemeleri önerilir.

3. Sistem İzleme Araçları: Windows Görev Yöneticisi ve Performans İzleyicisi gibi araçlar, bellek kullanımını izlemek ve potansiyel sorunları tespit etmek için kullanılabilir.



Şekil 12: Sanal bellek yönetimi örnek şeması.

KAYNAKÇA:

- [1] Sacco G. M., Schkolnick M. (1982): “A Mechanism for Managing the Buffer Pool in a Relational Database System Using the Hot Set Model”, In Proceedings of the 8th International Conference on Very Large DataBases, sf. 257–262, Mexico City, Meksika
- [2] Peter J. D. (1970): “Virtual Memory”, Computing Surveys, Cilt 2, No. 3, sf. 120-135.
- [3] Stefan M., Peter B. (2002): “Optimizing Main-Memory Join on Modern Hardware”, IEEE transactions on knowledge and data engineering, Cilt 14, No. 4, sf. 210-220.
- [4] İnternet: TBD Kamu Bilgi İşlem Merkezleri Yöneticileri Birliği (TBD Kamu BDB), “Sanallaştırma”, TBD Kamu-BDB Kamu Bilişim Platformu XII, http://www.tbd.org.tr/usr_img/cd/kamubib12/raporlarPDF/RP1-2010.pdf, 7,13-16, (2010).
- [5] Lunsford, D. L., “Virtualization Technologies in Information Systems Education”, Journal of Information Systems Education (JISE), Volume 20, Article 3, 339-348, (2009)
- [6] M. Suetake, T. Kashiwagi, H. Kizu, and K. Kourai, “S-memV: Split Migration of Large-memory Virtual Machines in IaaS Clouds,” in Proc. Int. Conf. Cloud Computing, 2018, pp. 285–293.
- [7] <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10197098>
- [8] <https://arxiv.org/abs/1510.03362>
- [9] . Borodin, A., El-Yaniv, R.: Online computation and competitive analysis. Cambridge University Press, New York, NY, USA (1998)
- [10] Edward G. Coffman, Jr. and Peter J. Denning Operating Systems Theory. Prentice-Hall, 1973

[11] <https://bilgisayarkavramlari.com/2009/05/29/sayfa-degistirme-algoritmasi-page-replacement/#4>

[12] <https://web.archive.org/web/20201025165312/https://www.hurriyet.com.tr/teknoloji/ssd-nedir-nasil-takilir-ve-calisir-41408909>