(CS-GY6233) Assignment 7

a)

(1 point) List three main differences between paging and segmentation

- Paging has no external fragmentation because all memory has been cut into frames. Frames do not have to be contiguous when being used by a process. Segmentation has no internal fragmentation because we allocate excactly how much we need. In Paging, we have “page” as the smallest unit, while a segment can be arbitrary size (in unit of byte, though).
- Paging uses a page table with TLB. Segmentation uses Segment tables and do relatively simple (compared with Paging) address translation and bound check by hardware. Segmentation has no such things as page fault, while Paging allow some virtaul addresses to refer to frames not in physical memory.
- Paging can do more optimization: putting only some part of virtual memory in physical memory, and copy-on-wirte.
- Paging needs more time to look up since virtual memory space (large table to look up) is larger than the one in Segmentation and since the possibility of page fault.

b)

(1 point) In memory paging, why must the page sizes be powers of 2?

Because we directly use address bits for page indecise and offsets. Left bits as page index and right bits as offset. So number of addresses in a page in 2^(number of bytes represented by offset bit).

c)

(1 point) In a computer system that supports programs of 512 pages, 128 bytes each, what is the minimum number of bits that can represent virtual addresses of such system?

512=2^9

128=2^7

ppppppppp bbbbbbb

ans: 9 + 7 = 16

d)

(1 point) In a 32-bit computer system (i.e. has 32 bits of virtual address space) and 1 KB page size, what is the page number of address 74373?

1KB = 2^10 Bytes

74373(10) = 10010001010000101 (2)

```
22                    10
ppppppppppppppppppppppp bbbbbbbbbb
               1001000 1010000101
```

ans: 1001000(2) = 72(10)

e)

(1 point) In a 32-bit computer system and 8-KB page size, if a process' size is 2 MB, how many entries are in the page table for such process?

8 KB = 2^13 bytes
offset = 19 bits
page size = 2^13 bytes
2 MB = 2^21 bytes

ans: 21 - 13 = 8 pages

f)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Dec  1 17:14:17 2020

n >=16 : do access n times
k >=3: page index: 0 ~ 2^k-1
f: 4 ~ 2^k: size of page table

@author: y56
"""

# random.randint(a, b)
#  Return a random integer N such that a <= N <= b

import random

do=True
while do:
    def fun(n,k):
        trace =  [random.randint(0,2**k-1) for _ in range(n)]
        miss_record=[-1]*(2**k+1)
        for table_size in range(4,2**k+1): # size of page table
            table=[]
            miss=0
            for ask in trace:
                if ask in table:
                    None
                else:
                    miss+=1
                    if len(table) < table_size:
                        table.append(ask)
                    else:
                        del table[0]
                        table.append(ask)
            miss_record[table_size]=miss
        return miss_record

    k=4
    miss_record=fun(64,k)

    import matplotlib.pyplot as plt
    plt.figure()
    plt.plot(range(4,2**k+1), miss_record[4:2**k+1])
    for left,right in zip(miss_record[4:2**k+1-1],miss_record[4+1:2**k+1]):
        if right>left:
            do=False
```

Spyder (Python 3.6)

File Edit Search Source Run Debug Consoles Projects Tools View Help

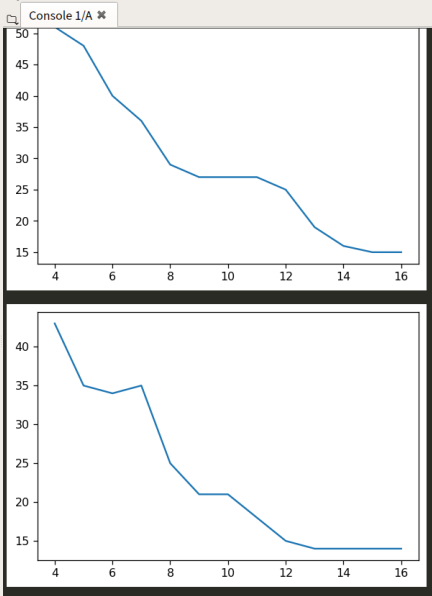
Editor - /home/y56/os2020fallcsgy6233/lab7/hw7_f.py

temp.py | untitled14.py | untitled15.py | hw7_f.py

```
# -*- coding: utf-8 -*-
"""
Created on Tue Dec  1 17:14:17 2020

n >=16 : do access n times
k >=3: page index: 0 ~ 2^k-1
f: 4 ~ 2^k: size of page table

@author: y56
"""

# random.randint(a, b)
#  Return a random integer N such that a <= N <= b

import random

do=True
while do:
    def fun(n,k):
        trace =  [random.randint(0,2**k-1) for _ in range(n)]
        miss_record=[-1]*(2**k+1)
        for table_size in range(4,2**k+1): # size of page table
            table=[]
            miss=0
            for ask in trace:
                if ask in table:
                    None
                else:
                    miss+=1
                    if len(table) < table_size:
                        table.append(ask)
                    else:
                        del table[0]
                        table.append(ask)
            miss_record[table_size]=miss
        return miss_record

    k=4
    miss_record=fun(64,k)

    import matplotlib.pyplot as plt
    plt.figure()
    plt.plot(range(4,2**k+1), miss_record[4:2**k+1])
    for left,right in zip(miss_record[4:2**k+1-1],miss_record[4+1:2**k+1]):
        if right>left:
            do=False
```

IPython console

Console 1/A




```
In [19]: ls
ans.md  ans.pdf  hw7_f.py  hw_7.png
Screenshot-20201201175753-1818x900.png

In [20]:
```

IPython console | History log | Help | Variable explorer | File explorer | Profiler

Permissions: RW End-of-lines: LF Encoding: UTF-8 Line: 9 Column: 1 Memory: 58 % CPU: 24 %