CSE150-F19-Homework2

Your NAME: \_\_\_Necole Goodman\_\_\_\_\_\_\_\_ Lab Section: \_\_\_\_\_\_\_\_\_

TA NAME: \_\_\_\_\_\_Heelam\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**1. Scheduling**

**a.** Here is a table of processes and their associated arrival and running times.

**Process ID Arrival Time CPU Running Time**

**P1** 0 6

**P2** 1 1

**P3** 4 2

**P4** 7 4

**P5** 8 3

Show the scheduling order for these processes under 3 policies: First Come First Serve (FCFS),

Shortest-Remaining-Time-First (SRTF), Round-Robin (RR) with timeslice quantum = 1. Assume

that context switch overhead is 0 and that new processes are added to the ***head*** of the queue except

for FCFS, where they are added to the ***tail***.

**Time FCFS SRTF RR**

0 1 1 1

1 1 2 2

2 1 1 3

3 1 1 4

4 1 3 5

5 1 3 1

6 2 1 3

7 3 1 3

8 3 1 5

9 4 5 1

10 4 5 4

11 4 5 5

12 4 4 1

13 5 4 4

14 5 4 1

15 5 4 1

CSE 150 Homework #2

**b.** For each process in each schedule above, indicate the queue wait time and completion time (otherwise

known as turnaround time, TRT). Note that wait time is the total time spend waiting in queue (all the

time in which the task is not running), while TRT is the total time from when the process arrives in the

queue until it is completed

**Scheduler P1 P2 P3 P4 P5**

**FCFS wait**

**FCFS TRT**

**SRTF wait**

**SRTF TRT**

**RR wait**

**RR TRT**

**c.** The SRTF algorithm requires knowledge of the future. Why is that? Name two ways to approximate the

information required to implement this algorithm.

2

CSE 150 Homework #2

**Caching and TLBs**

**a)** Which kind of locality makes large cache lines a good idea, and why?

Large cache lines are good when memory locations near to the current memory location being taken from the past and accessed in the future. Large cache lines are also good for providing spatial locality.

**b)** Consider a memory system employing 2 levels of cache with the following parameters.

**HL1** L1 cache hit rate 0.2

**HL2** L2 cache hit rate 0.5

**TL1** L1 cache access time 10ns

**TL2** L2 cache access time 100ns

**TM** Memory access time 2μs

Data access is attempted at L1, L2 and memory, in that order. Also, the miss time at each level includes

access times of previous levels (that is, if data is not in L1 cache, it will take 110ns to get it from the L2

cache, and so on).

**i.** What is the average (or effective) access time? *Fractional answers are okay*.

0.2 \*10 + (1 – 0.2) \* [0.5 \* (10+100) + (1 – 0.5) \* (10 + 100 + 2000)] = 890ns

**ii.** Assume a variation of this memory system that does not contain the L2 cache. This means the data is

either cached at L1 or in memory (all associated parameters stay the same). What is the average (or

effective) access time? *Fractional answers are okay*.

0.2 \* 10 + (1 – 0.2) \* ( 10 + 2000) = 1610 ns

3

Kernel Only

Uncacheable

0

0

Dirty

Use

Write

Valid

CSE 150 Homework #2

**Address Translation**

Consider a two-level memory management scheme on 20-bit virtual addresses using the following format for

virtual addresses:

Virtual Seg # Virtual Page # Offset

(2 bits) (6 bits) (12 bits)

Virtual addresses are translated into 20-bit physical addresses of the following form:

Physical Page # Offset

(8 bits) (12 bits)

Page table entries are 16 bits in the following format, stored in *big-endian form* in memory (i.e. the MSB is first

byte in memory).

**Page Table Entry (PTE)**

Physical Page #

(8 bits)

**a)** How big is a page? Explain.

A page is determined by the offset (12 bits). Page is 2^(offset) meaning 2^12, which is a product of 4096 bytes in size.

**b)** What is the maximum amount of virtual memory supported by this scheme? Explain.

The maximum amount of virtual memory is 2620 because virtual addresses are 20 bits.

**c)** What is the maximum amount of physical memory supported by this scheme? Explain.

With similar reasoning for the previous answer, physical memory can be calculated as 2^20 because physical addresses are 20 bits each.

4

CSE 150 Homework #2

**d)** Use the Segment Table and Physical Memory table given on the next page to predict what will happen

with the following load/store instructions. Addresses are virtual. The return value for a load is an 8-bit

data value or an error, while the return value for a store is either “**ok**” or an error. If there is an error,

make sure to say which error. Possibilities are: “**bad segment**” (invalid segment), “**segment overflow**”

(address outside range of segment), or “**access violation**” (page invalid, or attempt to write a read only

page). A few answers are given:

**Load[0xD2002] = 0x10, Store [0xD2031]=Access Violation,**

**Instruction Result Instruction Result**

**Load [0xC1015] 0x57 Store [0x52002] Segment overflow**

**Store [0x43045] ok Load [0x04013] 0x44**

**Store [0xC1016] Access violation Store [0x81015] Bad Segment**

**Load [0xD2002] 0x10 Store [0x03010] OK**

**Store [0xD2031] Access Violation Load [0x13035] 0x59**

**Virtual Address Format**

Page Table Max Page Segment State

Seg# Base Entries

0 0x02030 0x20 Valid

1 0x01020 0x10 Valid

2 0x01040 0x40 Invalid

3 0x04000 0x20 Valid

**Physical Memory**

**Address +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F**

**0x00000** 0E 0F 20 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D

**0x00010** 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D

**…**

**0x01010** 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F

**0x01020** 40 03 41 01 30 01 31 03 00 03 00 00 00 00 00 00

**0x01030** 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF

**0x01040** 10 01 11 03 31 03 13 00 14 01 15 03 16 01 17 00

**…**

**0x02030** 10 01 11 00 12 03 67 03 11 03 00 00 00 00 00 00

**0x02040** 02 20 03 30 04 40 05 50 01 60 03 70 08 80 09 90

**0x02050** 10 00 31 01 10 03 31 01 12 03 30 00 10 00 10 01

**…**

**0x04000** 30 00 31 01 11 01 33 03 34 01 35 00 43 38 32 79

**0x04010** 50 28 84 19 71 69 39 93 75 10 58 20 97 49 44 59

**0x04020** 23 03 20 03 00 01 62 08 99 86 28 03 48 25 34 21

5

CSE 150 Homework #2

**…**

**0x10000** AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55

**0x10010** A5 5A A5 5A A5 5A A5 5A A5 5A A5 5A A5 5A A5 5A

**…**

**0x11000** 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF

**0x11010** 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00

**0x11020** 22 33 44 55 66 77 88 99 AA BB CC DD EE FF 00 11

**…**

**0x31000** 01 12 23 34 45 56 67 78 89 9A AB BC CD DE EF 00

**0x31010** 02 13 24 35 46 57 68 79 8A 9B AC BD CE DF F0 01

**0x31020** 03 01 25 36 47 58 69 7A 8B 9C AD BE CF E0 F1 02

**0x31030** 04 15 26 37 48 59 70 7B 8C 9D AE BF D0 E1 F2 03

**…**

6