Stepping into the unknown

0.75 second, 1 megabytes

หลังจากที่เจ้าวัวน้อยได้ออกมาจากฟาร์มของคุณลุงใจดี เจ้าวัวได้เดินทางข้ามวันข้ามคืนจนในที่สุดก็ถึงสถานีรถไฟ เจ้าวัวต้องการเดินออกนอกเมืองเพื่อเริ่มต้นชีวิตใหม่ อยากพบเจออะไรที่แปลกใหม่ และไม่ใช่คนเดิมๆ อะไรเดิมๆ แต่เจ้าวัวก็ไม่มีตั๋วขึ้นรถไฟดังนั้นเจ้าวัวที่อยากขึ้นมากจึงไปอ้อนวอนเจ้าหน้าที่ประจำสถานี และเจ้าหน้าที่ก็เกิดใจอ่อนให้กับความน่ารักของเจ้าโชค จึงได้ยื่นข้อเสนอว่าถ้าตอบคำถามได้จะให้ตั๋วขึ้นรถไฟ

เนื่องจากเจ้าหน้าที่เรียนจบ Computer Science มาจึงคิดคำถามที่เกี่ยวข้องเพื่อมาถามเจ้าวัว โดยเจ้าหน้าที่จะให้เจ้าวัวช่วยคำนวณประสิทธิภาพของ Machine เครื่องหนึ่งชื่อ M1 โดยที่มี Instruction Set แบบ more64 ซึ่งมี Instruction Table อย่างง่ายตามตัวอย่างดังนี้

more64 Assembly	Description
JMP eax	One parameter command
ADD eax, ebx	Two parameters command
LEA eax, ebx, ecx*4, 8	Many parameters command
REBOOT (no parameter)	Restart the system
HALT (no parameter)	Shutdown the system

*กำหนดให้การอ่าน parameter เรียงลำดับจากซ้ายไปขวา

Machine M1 มี Main memory และ disk ที่ใช้จัดเก็บข้อมูลแบบ volatile และ non-volatile ตามลำดับ และกำหนดให้ระบบเป็น simple machine โดยจะละรายละเอียดเรื่อง architecture ออกไปและโฟกัสที่ memory โดยให้ cache และ Main memory เป็นอุปกรณ์เดียวกัน และเริ่มต้น machine จะไม่มีการใช้งาน memory และไม่มีข้อมูล store อยู่ใน disk โดยเก็บข้อมูลด้วยโครงสร้าง linked list และมีการ implement cache replacement policy ด้วย LRU linked list อย่างง่าย โดย Machine M1 มีระบบการทำงานโปรแกรมแบบขนานควบคู่ไปกับการทำ replacement policy โดยสอง process นี้ไม่เกี่ยวข้องกันซึ่ง Machine M1 มีอัตราการทำงาน n instructions/second และทุกๆการทำงาน instruction ครบ 1 วินาทีระบบ cache replacement policy จะคัดเลือก memory ที่มีอัตราส่วนการใช้งานต่อเวลา frequency/time (current – first access time) น้อยที่สุดออกไปและคัดเลือก memory ที่มีการใช้งานมากสุดจาก disk ขึ้นมาแทนที่ และ disk

memory ที่นำมาแทนที่จะต้องมีอัตราส่วนการใช้งานต่อเวลาที่ดีกว่า หากไม่จะแสดงผลบอกว่า Cache is better และหากการค้นหามี memory ไม่เพียงพอจะไม่ทำ replacement โดย memory ที่ถูกนำเข้ามายังระบบ cache จะถูกลบออกจาก disk และกลับเข้ามาเมื่อใช้งานเสร็จแล้ว

LRU Linked list อย่างง่ายมีการทำงานดังนี้

1. GET หาข้อมูล
   1.1. กรณีเจอ (cache hit) จะขยับ memory ขึ้นมาข้างหน้า
   1.2. กรณีไม่เจอ (cache miss) จากหาข้อมูลจาก disk
      1.2.1. หาก disk ไม่มีข้อมูลให้สร้าง memory ขึ้นมาใหม่
      1.2.2. เมื่อพบข้อมูลใน disk จะย้ายข้อมูลออกมาแล้ว PUT เข้า cache
2. PUT เพิ่มข้อมูล
   2.1. กรณีมีอยู่ใน cache สูงสุดจะนำส่วนท้ายออกไปใส่ disk จนมีเหลือเพียงพอ
   2.2. เมื่อมีพื้นที่ใน cache เพียงพอจะ PUT เข้าไปข้างหน้า

เจ้าหน้าที่ต้องการวัดประสิทธิภาพของการจัดการ cache ของระบบโดยจะมีการวัด cache hit rate และ cache miss rate และระยะเวลาการทำงาน ตลอดการทำงานของเครื่องนี้ โดยแต่ละคำสั่งการทำงานมี cost ในการทำงานไม่เท่ากัน เนื่องจากนิสิตเรียน CS จงช่วยเจ้าวัวตอบคำถามแสดงการทำงานของ Machine M1 และวัดประสิทธิภาพให้ถูกต้อง

ข้อกำหนดเพิ่มเติม

ไม่อนุญาตให้นิสิตใช้ C++ STL list, vector, array หรือ Data Structure อื่น ๆ นอกเหนือจาก Linked List

ข้อมูลนำเข้า

- รับค่า n_s เป็นขนาดของ cache และ n_a เป็นจำนวน assembly instructions ที่ machine รองรับ คั่นด้วยช่องว่าง โดย $n_s, n_a \in \mathbb{Z}^+ ; 0 \leq n_s, n_a$
- รับค่า n_i โดย $n_i \in \mathbb{Z}^+ ; 0 < n_i$ เป็นจำนวน instructions ต่อ 1 วินาทีของ Machine M1
- รับค่า t_{hit} และ $t_{miss} \in \mathbb{R}^+$ ในหน่วยวินาทีเป็น cost ของ cache hit และ cache miss ตามลำดับขั้นด้วยช่องว่าง n_a บรรทัด
- รับค่า a_i, p_i และ t_i เป็นคำสั่ง assembly, จำนวน parameter และ cost เวลาการทำงาน ตามลำดับขั้นด้วยช่องว่าง จำนวน
  - โดย a_i เป็น non-empty string, $p_i \in \mathbb{Z}^+$และ $t_i \in \mathbb{R}^+$

ข้อมูลส่งออก

- แสดงผลการทำงานของระบบ cache replacement และ disk memory
- แสดงผลค่าสถานะของ cache และ disk ภายหลังการทำงานเสร็จสิ้น
- แสดงผลเวลาทั้งหมดที่ใช้หน่วยวินาที, จำนวน cache hit และจำนวน cache miss
- หากไม่สามารถแสดงผลได้ให้แสดงผล IMPOSSIBLE

ตัวอย่างข้อมูลนำเข้าและข้อมูลส่งออก (Input & Output Examples)

ข้อมูลนำเข้า	ข้อมูลส่งออก
1 2 10 0.1 1.0 JMP 1 0.2 CALL 1 0.3 JMP eax CALL ebx JMP eax	---------- MOVE JMP FROM DISK PUT JMP GET (MISS) JMP FIRST ACCESS : 0.000s ---------- MOVE eax FROM DISK MOVE JMP TO DISK PUT eax GET (MISS) eax FIRST ACCESS : 0.000s ---------- MOVE CALL FROM DISK MOVE eax TO DISK PUT CALL GET (MISS) CALL FIRST ACCESS : 0.200s ---------- MOVE ebx FROM DISK MOVE CALL TO DISK PUT ebx GET (MISS) ebx FIRST ACCESS : 0.200s ---------- MOVE JMP FROM DISK MOVE ebx TO DISK PUT JMP GET (MISS) JMP ---------- MOVE eax FROM DISK MOVE JMP TO DISK PUT eax GET (MISS) eax ========== (REPORT) LRU List : eax <- NULL Disk List : CALL <- ebx <- JMP <- NULL Total Time : 6.700s CacheHit : 0 CacheMiss : 6

0 2 10 0.1 1.0 ADD 0 0.5 SUB 0 0.3 ADD SUB	IMPOSSIBLE
3 3 10 0.1 1.0 NOP 0 0.1 WAIT 0 0.2 HALT 0 0.0 NOP WAIT HALT	---------- MOVE NOP FROM DISK PUT NOP GET (MISS) NOP FIRST ACCESS : 0.000s ---------- MOVE WAIT FROM DISK PUT WAIT GET (MISS) WAIT FIRST ACCESS : 0.100s ---------- MOVE HALT FROM DISK PUT HALT GET (MISS) HALT FIRST ACCESS : 0.300s ========== (REPORT) LRU List : HALT <- WAIT <- NOP <- NULL Disk List : <- NULL Total Time : 3.300s CacheHit : 0 CacheMiss : 3

2 3
10
0.1 1.0
LEA 4 0.8
ADD 2 0.4
MUL 3 0.6
LEA eax, ebx, ecx, edx
ADD eax, ebx
MUL eax, ebx, ecx
	---------- 	
	MOVE LEA FROM DISK
	PUT LEA
	GET (MISS) LEA
	FIRST ACCESS : 0.000s
	---------- 	
	MOVE eax FROM DISK
	PUT eax
	GET (MISS) eax
	FIRST ACCESS : 0.000s
	---------- 	
	MOVE ebx FROM DISK
	MOVE LEA TO DISK
	PUT ebx
	GET (MISS) ebx
	FIRST ACCESS : 0.000s
	---------- 	
	MOVE ecx FROM DISK

```
MOVE eax TO DISK
PUT ecx
GET (MISS) ecx
FIRST ACCESS : 0.000s
------------------------------
MOVE edx FROM DISK
MOVE ebx TO DISK
PUT edx
GET (MISS) edx
FIRST ACCESS : 0.000s
------------------------------
MOVE ADD FROM DISK
MOVE ecx TO DISK
PUT ADD
GET (MISS) ADD
FIRST ACCESS : 0.800s
------------------------------
MOVE eax FROM DISK
MOVE edx TO DISK
PUT eax
GET (MISS) eax
------------------------------
MOVE ebx FROM DISK
MOVE ADD TO DISK
PUT ebx
GET (MISS) ebx
------------------------------
MOVE MUL FROM DISK
MOVE eax TO DISK
PUT MUL
GET (MISS) MUL
FIRST ACCESS : 1.200s
------------------------------
MOVE eax FROM DISK
MOVE ebx TO DISK
PUT eax
GET (MISS) eax
------------------------------
MOVE ebx FROM DISK
MOVE MUL TO DISK
PUT ebx
GET (MISS) ebx
------------------------------
MOVE ecx FROM DISK
MOVE eax TO DISK
PUT ecx
GET (MISS) ecx
==============================
(REPORT)
LRU List : ecx <- ebx <- NULL
```

	Disk List : LEA <- edx <- ADD <- MUL <- eax <- NULL Total Time : 13.800s CacheHit : 0 CacheMiss : 12
2 3 3 0.1 1.0 FAST 0 0.1 SLOW 0 1.0 MED 0 0.5 FAST FAST FAST SLOW MED SLOW	---------- MOVE FAST FROM DISK PUT FAST GET (MISS) FAST FIRST ACCESS : 0.000s ---------- GET (HIT) FAST ---------- GET (HIT) FAST ========== (AUTO REPLACEMENT) NO REPLACEMENT ---------- MOVE SLOW FROM DISK PUT SLOW GET (MISS) SLOW FIRST ACCESS : 0.300s ---------- MOVE MED FROM DISK MOVE FAST TO DISK PUT MED GET (MISS) MED FIRST ACCESS : 1.300s ---------- GET (HIT) SLOW ========== (AUTO REPLACEMENT) SLOW(1.333) - Cache FAST(1.667) - Disk MOVE FAST FROM DISK MOVE SLOW TO DISK ========== (REPORT) LRU List : FAST <- MED <- NULL Disk List : SLOW <- NULL Total Time : 6.100s CacheHit : 3 CacheMiss : 3

2 3 2 0.1 1.0 A 0 0.2 B 0 0.2 REBOOT 0 0.0 A B A B REBOOT	---------- MOVE A FROM DISK PUT A GET (MISS) A FIRST ACCESS : 0.000s ---------- MOVE B FROM DISK PUT B GET (MISS) B FIRST ACCESS : 0.200s ========== (AUTO REPLACEMENT) NO REPLACEMENT ---------- GET (HIT) A ---------- GET (HIT) B ========== (AUTO REPLACEMENT) NO REPLACEMENT ---------- MOVE REBOOT FROM DISK MOVE A TO DISK PUT REBOOT GET (MISS) REBOOT FIRST ACCESS : 0.800s ========== (REBOOT) MOVE REBOOT TO DISK MOVE B TO DISK RESTORE B FROM DISK RESTORE REBOOT FROM DISK ========== (REPORT) LRU List : REBOOT <- B <- NULL Disk List : A <- NULL Total Time : 4.000s CacheHit : 2 CacheMiss : 3

จำนวนชุดทดสอบ: 31 ชุด