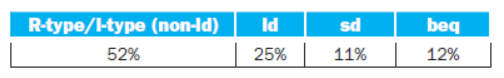
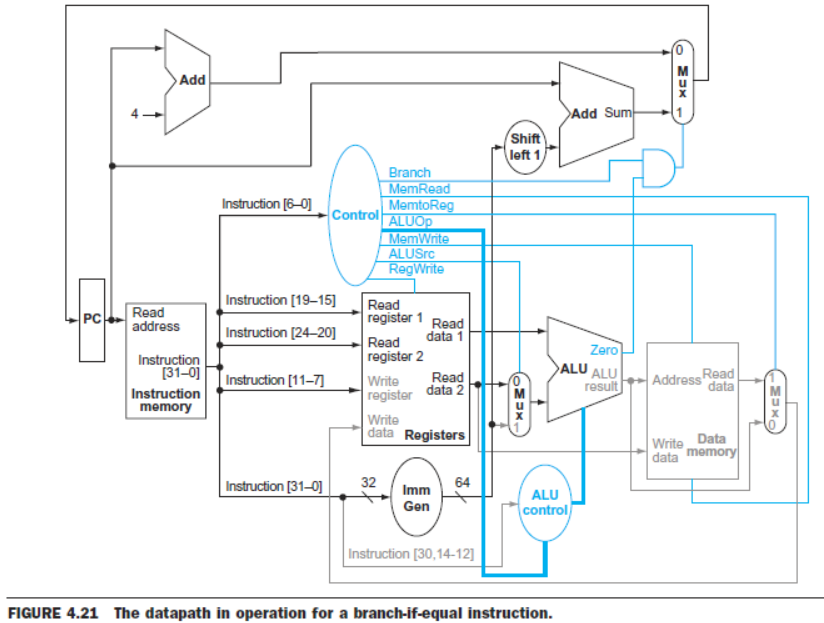
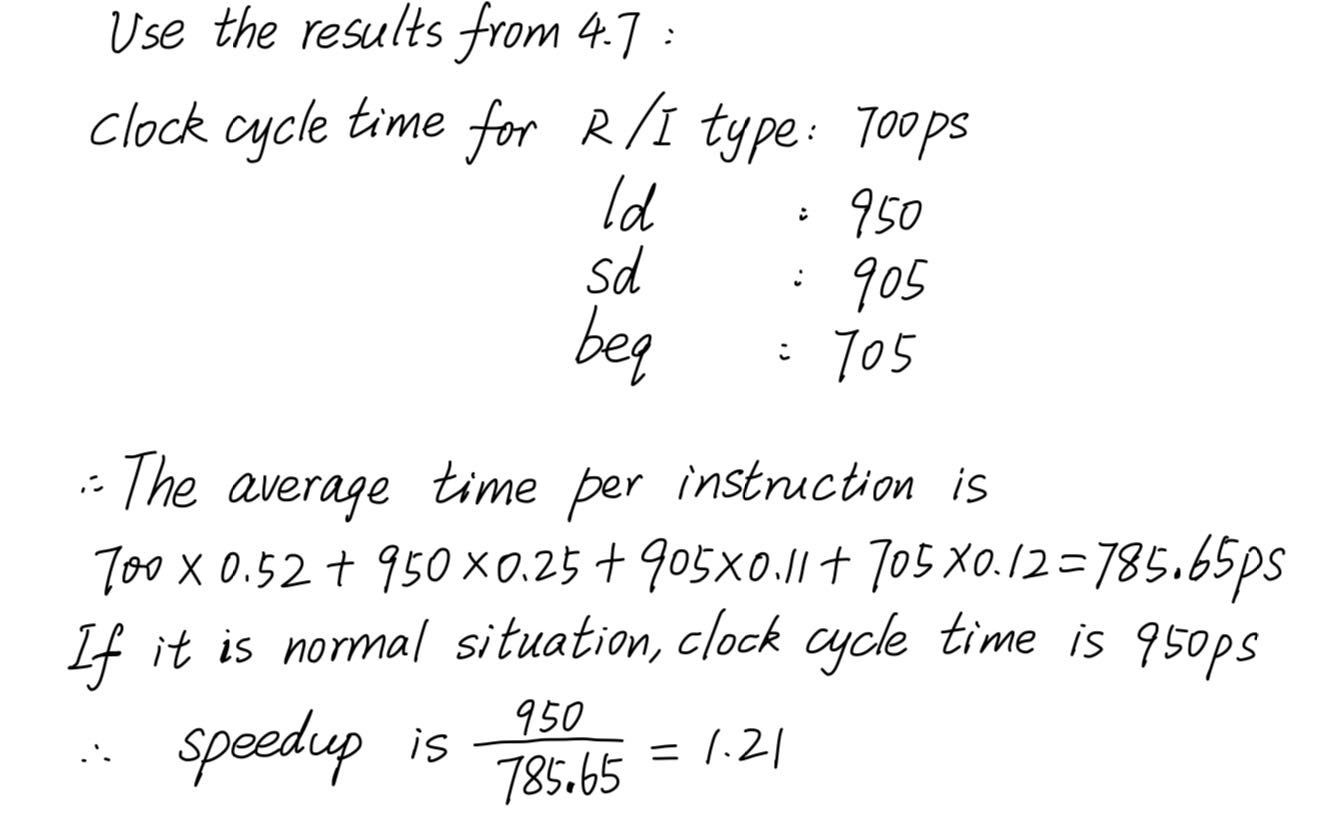
# Chapter 4

1. **【书Exercise 4.8】Suppose you could build a CPU where the clock cycle time was** different for each instruction. What would the speedup of this new CPU be over the CPU presented in Figure 4.21 given the instruction mix below?





****

1. **‬Suppose you executed the code below on a version of the pipeline CPU that** does not handle hazards. All registers are initialized to zero. Mem(1)=0xaa,Mem(9)=0xbb. The Branch takes place depending on the result of comparison in the stage of memory access.

main: addi x0,x0,0x0

addi x1,x0,0x1

addi x2,x0,0x1

addi x3,x0,0x1

addi x4,x0,0x1

lw x20,0x8(x1)

andi x4,x20,0x2

ori x3,x4,0x8

and x5,x6,x7

sw x7,0x1(x0)

xor x8,x3,x2

addi x9,x1,0x7

lw x9,0x1(x0)

and x10,x4,x3

beq x9,x8,loop1

addi x17,x8,0x2

add x11,x2,x4

loop1: addi x13,x13, 1

andi x15,x8,0x1

or x28,x9,x3

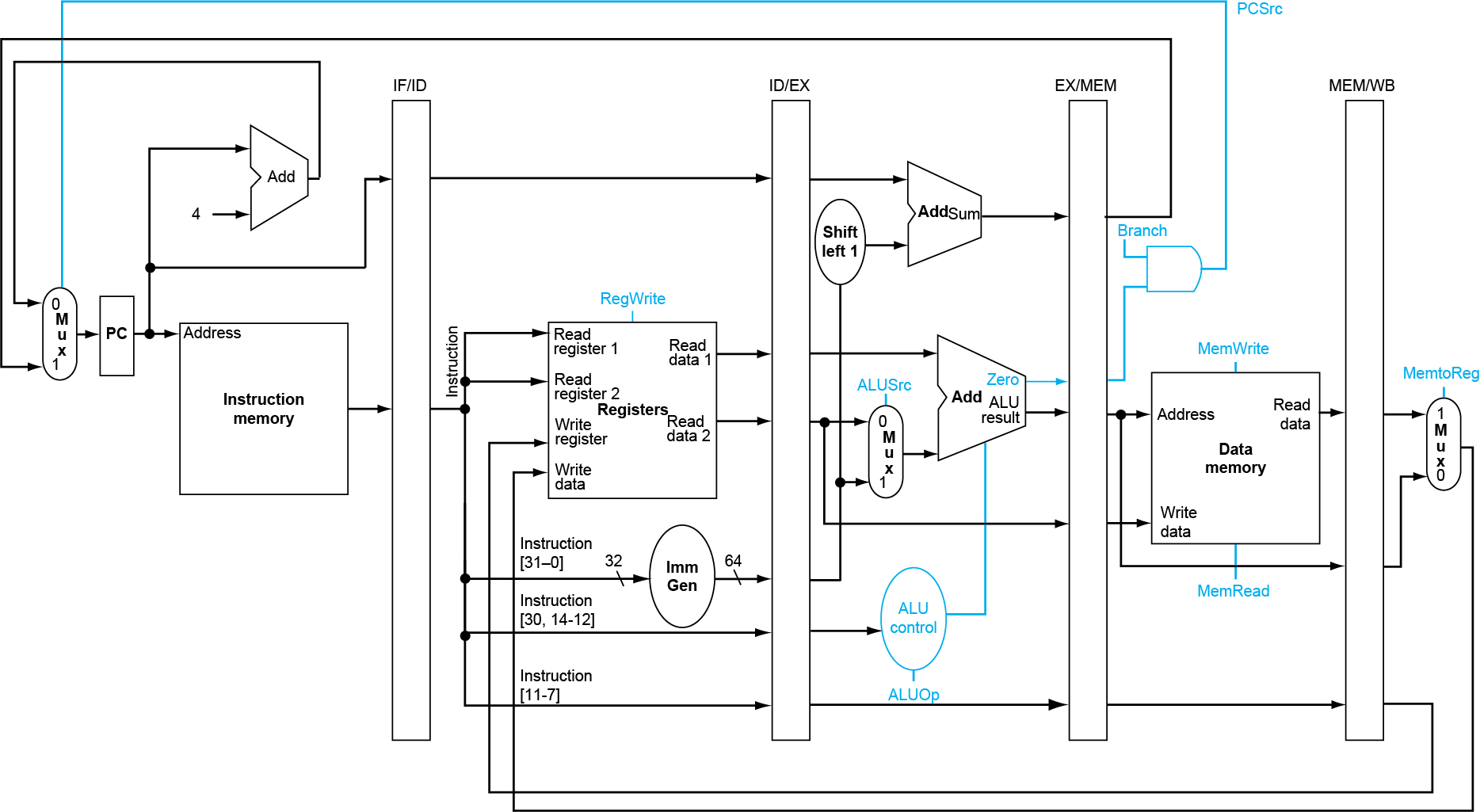
add x16,x10,x10

……………………

After the instructions running, the content of register x3 is ( 9 ),

X4 is ( 0 ),x8 is ( 8 ),x17 is ( 10 ),x13 is ( 1 ).

1. **Pipeline CPU Design**



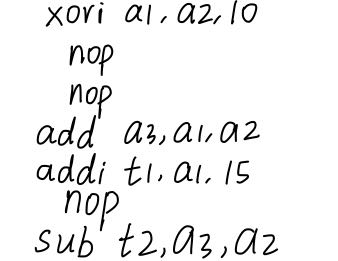
1) Assume that a1 is initialized to AA and a2 is initialized to CC. Suppose you executed the code below on a version of the pipeline CPU without solving data hazards. Now please adopt a measure of software(add nop) to make the pipeling run correctly.

xori a1, a2, 10

add a3, a1, a2

addi t1, a1, 15

sub t2, a3, a2

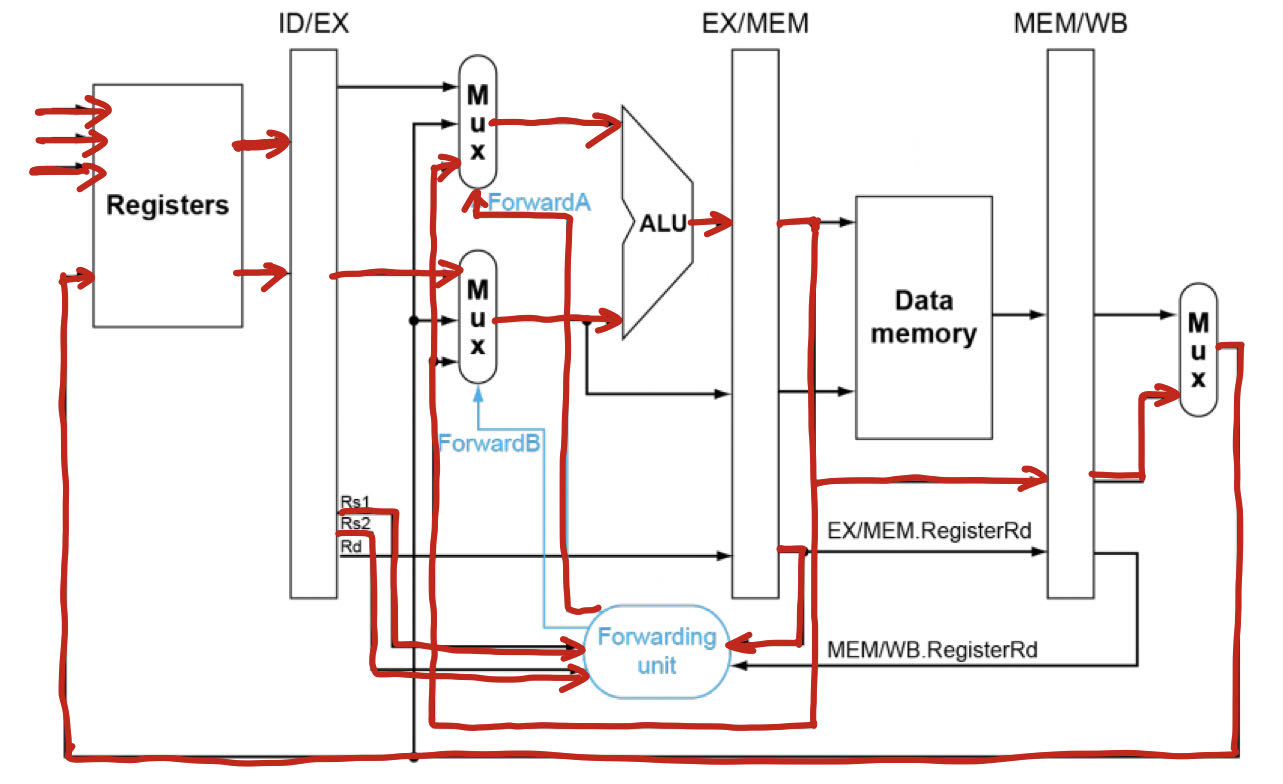


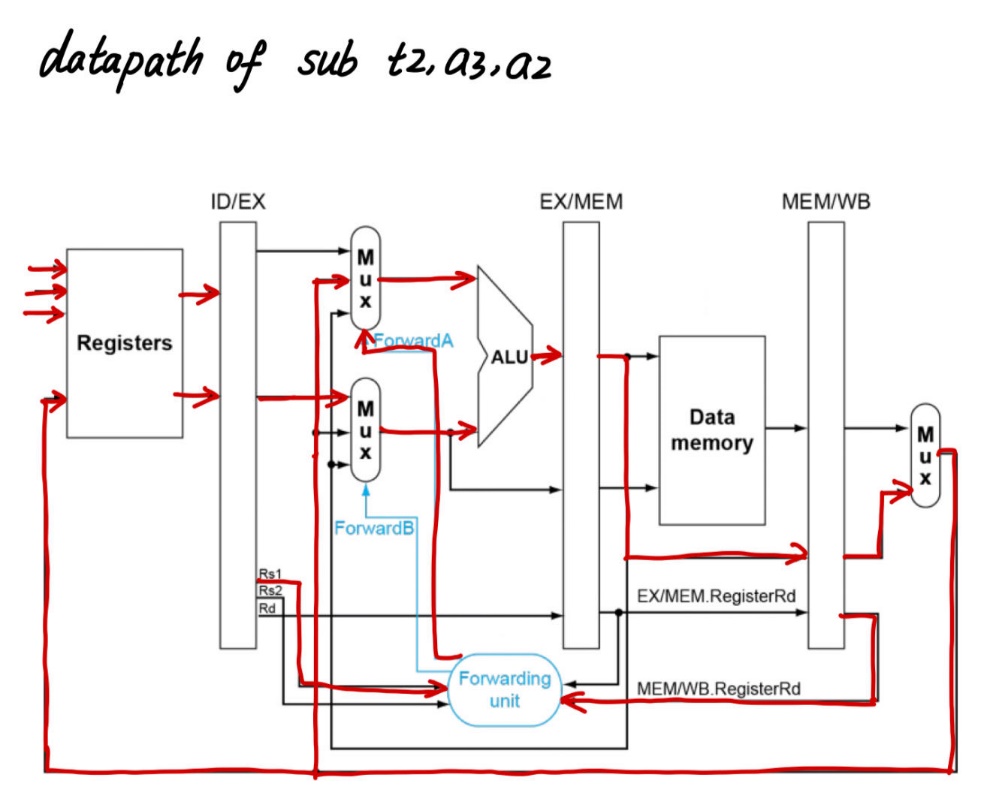
2) After you modify the code, please draw the corresponding instruction for each level of pipeline under the fourth clock.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IF | ID | EX | MEM | WB |
| add a3,a1,a2 | nop | nop | xori a1,a2,10 | \_\_\_ |

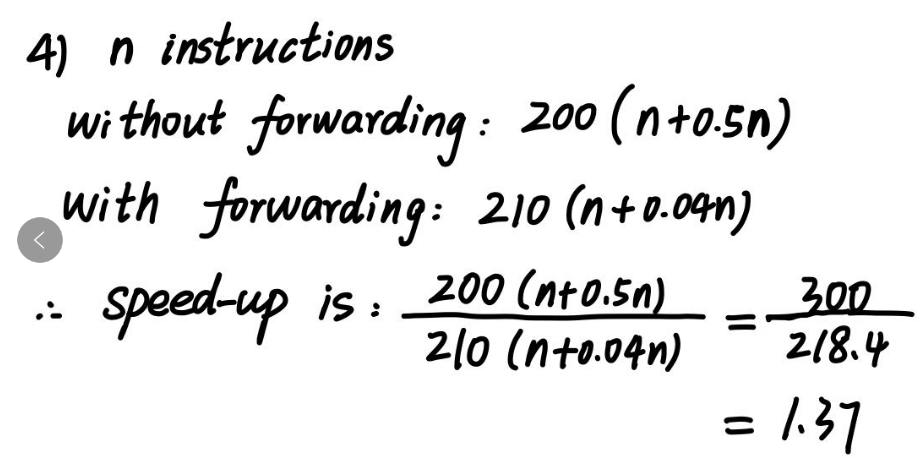
3) Now please adopt the measure of forwarding to deal with the data harzard. You have to draw the datapath for the forwarding.

Datapath of add a3,a1,a2



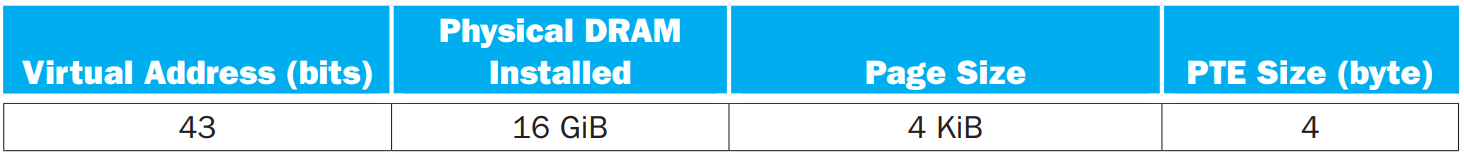


4) Suppose that the cycle time of this pipeline without forwarding is 200 ps. Suppose also that adding forwarding hardware will reduce the number of NOPs from 0.5\*n to 0.04\*n, but increase the cycle time to 210 ps. What is the speed up of this new pipeline compared to the one without forwarding?



# Chapter 5

1. **书后5.18 In this exercise, we will examine space/time optimizations for page** tables. The following list provides parameters of a virtual memory system.



1. For a single-level page table, 2^\_\_31\_\_\_(number) page table entries (PTEs) are needed. \_\_8\_\_GB physical memory is needed for storing the page table?
2. Using a multi-level page table can reduce the physical memory consumption of page tables by only keeping active PTEs in physical memory. \_\_2\_\_\_ levels of page tables will be needed if the segment tables (the upper-level page tables) are allowed to be of unlimited size.
3. Suppose the segments are limited to the 4 KiB page size (so that they can be paged). Is 4 bytes large enough for all page table entries (including those in the segment tables?( A )
4. Yes B. No
5. \_\_4\_\_\_\_ levels of page tables are needed if the segments are limited to the 4 KiB page size.
6. **A memory and cache subsystem uses byte-addressing, 32-bit address. Each row** of the table below indicates a kind of cache, the “Cache feature” column of table describes the total size of cache data (not containing tag and valid bit), cache kind, cache block size. A 32-bit memory address 0x22339AB contains 3 fields: tag, index, byte offset, some of these field’s size(bit number of this field) or value should be filled into table blank below. You should use Hex-decimal number to fill into column “Index value(Hex)”. Total cache size containing data block, tag and valid bit should also be filled into column “Tatal Size,KB”(KB means unit is KB—1024 byte, not byte)，only number (not expression) is allowed to be filled into this colomn.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Cache feature** | **Index value(Hex)** | **Index size, bits** | **TAG size, bits** | **Tatal Size,KB** |
| 64KB cache data,Direct-mapped,  8 bytes/block | 0x0735 | 13 | 16 | 81 |
| 512KB cache data, Direct-mapped,  64 bytes/block | 0x­­­ce6 | 13 | 13 | 526 |
| 512KB cache data, 2-Way set associative  8 bytes/block | 0x6735 | 15 | 14 | 632 |
| 1024KB cache data, 8-Way set associative  32 bytes/block | 0x9cd | 12 | 15 | **1088** |

Example value has been given in first question row.

1. **For a two-way set associate cache design with a 32-bit address, the following** bits of the address are used to access the cache.

|  |  |  |
| --- | --- | --- |
| Tag | Index | Offset |
| 31-11 | 10-4 | 3-0 |

1) What is the cache block size(in words，32bits/word)?\_\_ 4\_\_\_

2) How many blocks does the cache have? \_\_\_ 256 \_

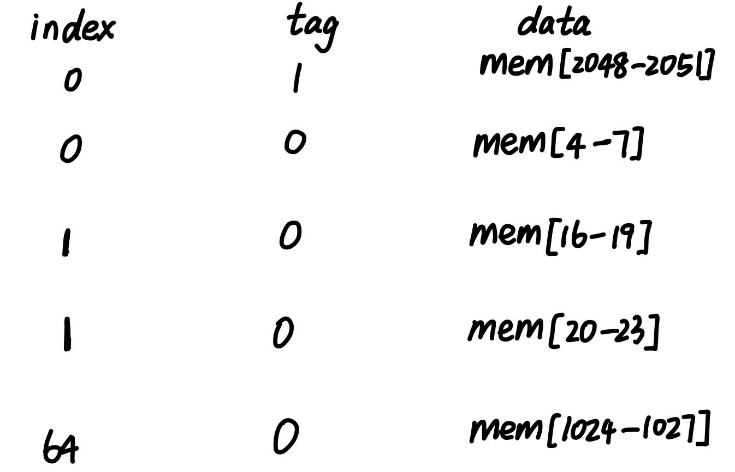
*3) The following byte-addressed cache reference are recorded.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *0* | *4* | *16* | *20* | *1024* | *2048* | *4* |

*How many blocks are replaced in LRU?* \_\_\_1

*What is the hit ratio \_ 1/7 \_\_*

*4) List the final state of the cache after 3), with each valid entry represented as a record of <index, tag, data>*

**