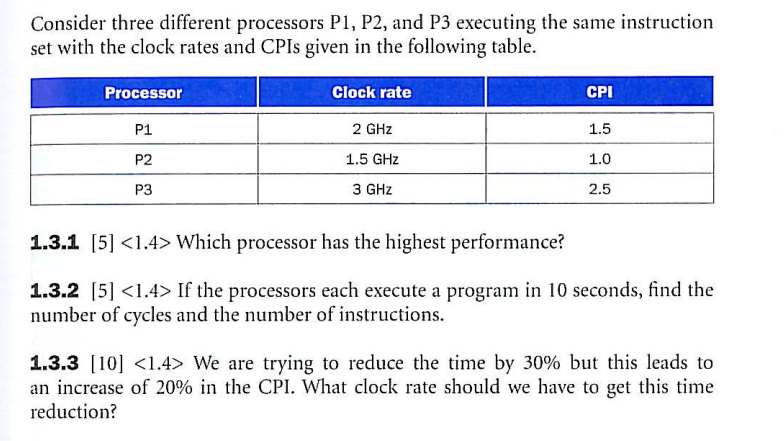
**Đề tổng tổng hợp dạng bài tập giữa kỳ KTMT**

1. Tính hiệu suất



**Dịch:** Các CPU ở trên thực thi cùng tập lệnh, hiệu năng của CPU nào cao nhất ; Nếu thời gian thực thi của mỗi CPU là 10s, hãy tìm số clock cycles và số dòng lệnh cho các CPU đó ; Nếu giảm 30% thời gian thực thi (lấy từ câu b) của mỗi CPU thì phải tăng CPI lên 20%. Tìm clock rate cần thiết cho mỗi CPU để thỏa mãn điều trên.

Gợi ý giải: Dùng công thức dễ dàng suy ra thời gian thực thi của 3 CPU , từ đó suy ra hiệu suất. Câu tiếp theo, có thời gian dễ dàng suy ra => clock cycles và instructions.

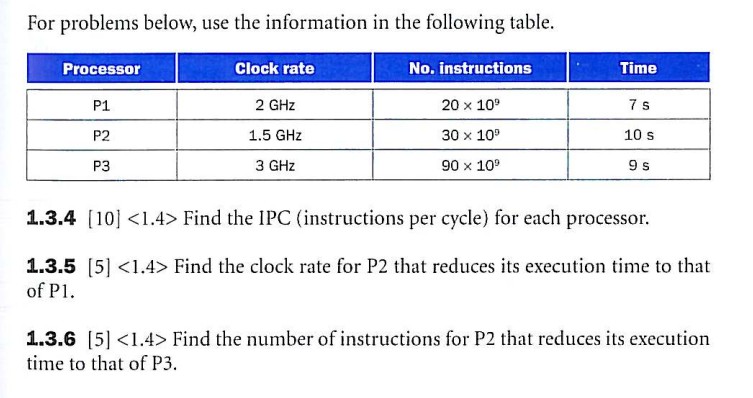
Câu cuối, ta có CPU time (new) = CPU time (old) - CPU time (old) \* 30% = CPU Time(old)\* 70%

CPI new = CPI old + CPI old \* 20% = CPI old \* 120%

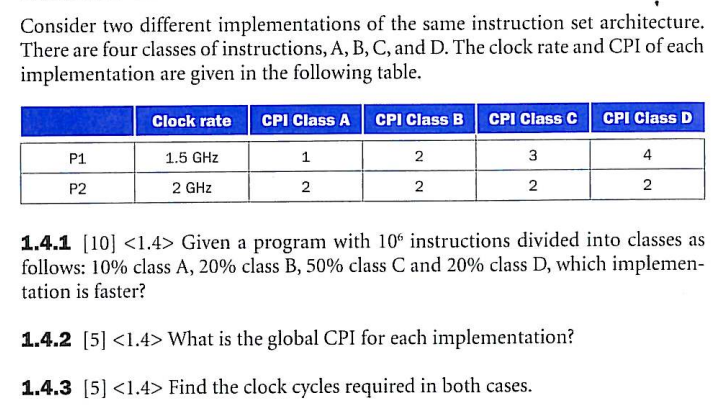
Instruction count = [CPU Time (old) \* Clock rate (old) ]/ CPI(old) = [CPU Time(old)\*70%\*Clock rate(new)] / [CPI(old)\*120%] (Vì thực hiện cùng 1 tập lệnh)

Với P1 => Clock rate mới = 3,42Ghz

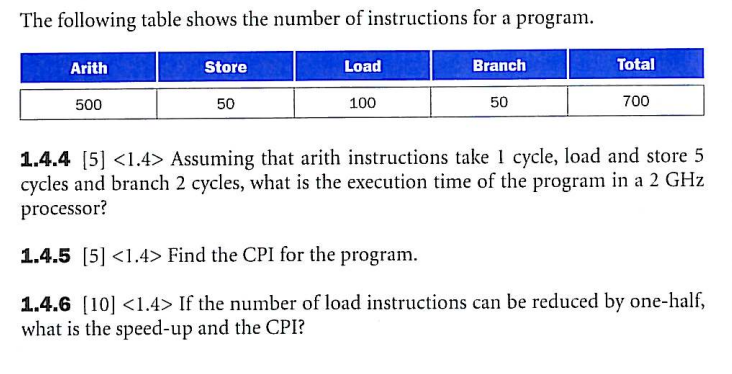
Các câu sau làm tương tự.



**Dịch:** Cho các thông số ở bảng trên, tìm IPC (Số dòng lệnh trên 1 clock cycle) của mỗi CPU ; Tìm clock rate của P2 để P2 có thời gian thực thi giống P1 ; Tìm số lệnh mà P2 thực thi nếu thời gian thực thi của P2 bằng thời gian thực thi của P3.



**Dịch:** 2 CPU P1 và P2 thực thi cùng một tập lệnh, tập lệnh được chia làm 4 lớp A, B, C, D và chiếm số dòng lệnh như trên. CPU nào có hiệu suất cao hơn? ; Tìm CPI trung bình cho mỗi CPU; Tìm số clock cycles cho mỗi CPU?



**Dịch:** Có một đoạn chương trình gồm 700 dòng lệnh và được chia làm 4 lớp Arith , Store , Load, Branch. Giả sử mỗi câu lệnh ở lớp Arith chiếm 1 clock cycle, lớp Load và lớp Store chiếm 5 clock cycles, lớp Branch chiếm 2 clock cycles. Tìm thời gian thực thi của CPU nếu CPU đó có clock rate là 2 Ghz; tìm CPI cho CPU đó; Nếu số dòng lệnh của lớp Load giảm một nửa thì thời gian thực thi và CPI cho CPU đó là bao nhiêu?

1. Chương II
2. Đổi câu lệnh C sang Assembly. Cho các biến i, j, h, g được lưu trên các thanh ghi $s0 -> $s3, địa chỉ nền của 2 mảng A, B được lưu trên thanh ghi $s4 và $s5.
3. i = h + g + A[5] ;
4. A[100-h] = B[3] + 6;
5. B[i-j] = A[B[i] – j] + h/256 ;
6. for(int i=8; i < 69; i++)

{

A[i+5] = i - j;

}

1. if( i>j )

{

i = i – j;

}

else

i = i + A[j-6];

1. while(i!=j)

{

if(i>j)

{

i = i – j;

}

else

j = j – i ;

}

1. while(i--) // i trừ sau, tức so sánh xong mới trừ

{

A[i-4] = B[i + j];

}

Đáp án câu này: Dòng lệnh trên tương đương với dòng lệnh dưới

while(i != 0)

{

i--;

A[i-4] = B[i + j];

}

Cái này các bạn tự làm nhé. ☺

Lưu ý: Gặp dạng này nhớ trừ ngay sau khi so sánh điều kiện dừng rồi thực hiện phần thân. Tương tự đối với câu lệnh dạng ++i ; --i; i++ !

1. Chuyển đổi mã Assembly sang mã máy và ngược lại.
2. Add $t0, $t1, $zero
3. Addi $s1, $s5, 96
4. Nor $t2, $s5, $s3
5. Lw $s0, 16($s3)
6. Sw $s1, 20($t4)

Chuyển các mã sau sang mã Assembly

1. 0000 0010 0001 0000 1000 0000 0010 0000
2. 0000 0001 0100 1011 0100 1000 0010 0010
3. 0101 1111 1011 1110 0100 0000 0000 0000