Σε αυτή την ερώτηση θα αναλύσουμε την απόδοση C σε πολυνηματική αρχιτεκτονική. Θα πρέπει να υποθέσετε ότι πίνακες A, B και Γ δεν επικαλύπτονται στη μνήμη.

Το μηχάνημά μας είναι ένας επεξεργαστής μονής ακολουθίας εκτέλεσης κατά παραγγελία. Μεταβαίνει σε διαφορετικό νήμα κάθε κύκλου χρησιμοποιώντας σταθερό χρονοπρογραμματισμό round robin. Καθένα από τα νήματα N εκτελούν μία εντολή κάθε N κύκλους. Κατακερματίζουμε τον κώδικα σε νήματα έτσι ώστε κάθε νήμα να εκτελεί κάθε Nη επανάληψη του αρχικού προγράμματος C.
Οι εντολές ακέραιων χρειάζονται 1 κύκλο για να εκτελεστούν, οι εντολές κινητής υποδιαστολής 4 κύκλους και οι εντολές μνήμης χρειάζονται 3 κύκλους. Όλες οι μονάδες εκτέλεσης είναι πλήρως διασωληνωμένος. Εάν μια εντολή δεν μπορεί να εκτελεστεί επειδή τα δεδομένα της δεν είναι ακόμα διαθέσιμα, εισάγει μια φυσαλίδα (ένα κενό) στον αγωγό και προσπαθεί ξανά μετά το N κύκλους.
Παρακάτω είναι το πρόγραμμά μας σε κώδικα μηχανής για αυτό το μηχάνημα για ένα μόνο νήμα που εκτελεί ολόκληρο τον βρόχο.

```
loop:  ld f1, 0(r1)      ; f1 = A[i]
       ld f2, 0(r2)      ; f2 = B[i]
       fmul f4, f2, f1   ; f4 = f1 * f2
       st f4, 0(r1)      ; A[i] = f4
       ld f3, 0(r3)      ; f3 = C[i]
       fadd f5, f4, f3   ; f5 = f4 + f3
       st f5, 0(r3)      ; C[i] = f5
       add r1, r1, 4     ; i++
       add r2, r2, 4
       add r3, r3, 4
       add r4, r4, -1
       bnez r4, loop     ; loop
```

α. Εκχωρούμε τον κώδικα του βρόχου σε N νήματα έτσι ώστε κάθε νήμα εκτελεί κάθε Nη επανάληψη του αρχικού βρόχου. Γράψτε τον κώδικα μηχανής που θα εκτελούσε ένα από τα N νήματα σε αυτή την πολυνηματική μηχανή.
β. Ποιος είναι ο ελάχιστος αριθμός νημάτων που χρειάζεται ώστε το μηχάνημα να παραμένει πλήρως αξιοποιημένο εκδίδοντας μια εντολή για κάθε κύκλο για το πρόγραμμα μας;
γ. Θα μπορούσαμε να φτάσουμε στην κορυφαία απόδοση χρησιμοποιώντας λιγότερα νήματα με αναδιάταξη των οδηγιών; Εξηγήστε συνοπτικά.
δ. Ποια είναι η μέγιστη θεωρητική απόδοση σε flops/κύκλο για αυτό το πρόγραμμα;

A)

```
loop:
ld f1, 0 (r1+(N-1)*4)
ld f2, 0 (r2+(N-1)*4)
fmul f4, f2, f1
st f4, 0 (r1+(N-1)*4)
ld f3, 0(r3+(N-1)*4)
fadd f5, f4, f3
st f5, 0(r3+(N-1)*4)
add r1, r1, 4*N
add r2, r2, 4*N
add r3, r3, 4*N
add r4, r4, -N
bnez r4, loop
```

B)

Σύμφωνα με την εκφώνηση η εκτέλεση εντολής πολλαπλασιασμού χρειάζεται 4 κύκλους για να ολοκληρωθεί, που την καθορίζει ως την πιο χρονοβόρα εντολή προς εκτέλεση επομένως ο ελάχιστος αριθμός νημάτων για την βελτιστοποίηση εκτέλεσης του προγράμματος θα είναι 4 καθώς ανά 4 κύκλους πάντα κάποιο νήμα θα είναι ελεύθερο για χρήση και έτσι δεν θα δημιουργηθεί καμιά φυσαλίδα στην διασωλήνωση.

Γ)

Εφαρμόζοντας αναδιάταξη θα μπορούσαμε να βελτιστοποιήσουμε την απόδοση του κώδικα και συγκεκριμένα αν εκτελέσουμε τις τελευταίες εντολές add πριν την fadd, θα εξαλείψουμε τους 4 κύκλους που χρειάζεται η fadd αφού θα εκτελεστούν παράλληλα

Τελικός κώδικας:

```
Loop:  ld f1, 0 (r1)

ld f2, 0 (r2)

fmul f4, f2, f1

st f4, f2, f1

ld f3, 0(r3)

add r1, r1, 4

add r2, r2, 4

add r3, r3, 4

add r4, r4, -1

fadd f5, f4, f3

st f5, 0(r3)

bnez r4, loop
```

Δ)

Οι συνολικοί κύκλοι είναι για το πρόγραμμα πριν την αναδιάταξη είναι 28 ενώ μετά είναι 24, οπότε με 4 νήματα η μέγιστη θεωρητική απόδοση flops/cycle είναι 4/24 = 1/6.