1. A processor with dynamic scheduling and issue bound operand fetch has 3 execution units – one LOAD/STORE unit, one ADD/SUB unit and one MUL/DIV unit. It has a reservation station with 1 slot per execution unit and a single register file. Starting with the following instruction sequence in the instruction fetch buffer and empty reservation stations, for each instruction find the cycle in which it will be issued and the cycle in which it will write result.
load R6, 34(R12)
load R2, 45(R13)
mul R0, R2, R4
sub R8, R2, R6
div R10, R0, R6
add R6, R8, R2
Assume out of order issue and out of order execution. Execute cycles taken by different instructions are:
LOAD/STORE : 2
ADD/SUB : 1
MUL : 2
DIV : 4

΄Ενας επεξεργαστής με δυναμικό χρονοπρογραμματισμό και πολιτική ανάκτησης τελεστών δεσμευμένη (issue-bound fetch policy) έχει 3 μονάδες εκτέλεσης – μία μονάδα LOAD/STORE, μία μονάδα ADD/SUB και μία μονάδα MUL/DIV. Διαθέτει σταθμό κρατήσεων (reservation station) με 1 θέση (slot) ανά μονάδα εκτέλεσης και ένα ενιαίο αρχείο καταχωρητών. Οι κύκλοι εκτέλεσης που απαιτούνται για κάθε τύπο εντολής είναι:
LOAD/STORE : 2
ADD/SUB : 1
MUL : 2
DIV : 4

α. Ξεκινώντας με την παρακάτω ακολουθία εντολών στην προσωρινή μνήμη ανάκτησης εντολών και άδειους σταθμούς κρατήσεων, για κάθε εντολή βρείτε τον κύκλο στον οποίο θα εκκινήσει (issue cycle) και τον κύκλο στον οποίο θα "γράψει" το αποτέλεσμα (result write cycle).
load R6, 34(R12)
load R2, 45(R13)
mul R0, R2, R4
sub R8, R2, R6
div R10, R0, R6
add R6, R8, R2

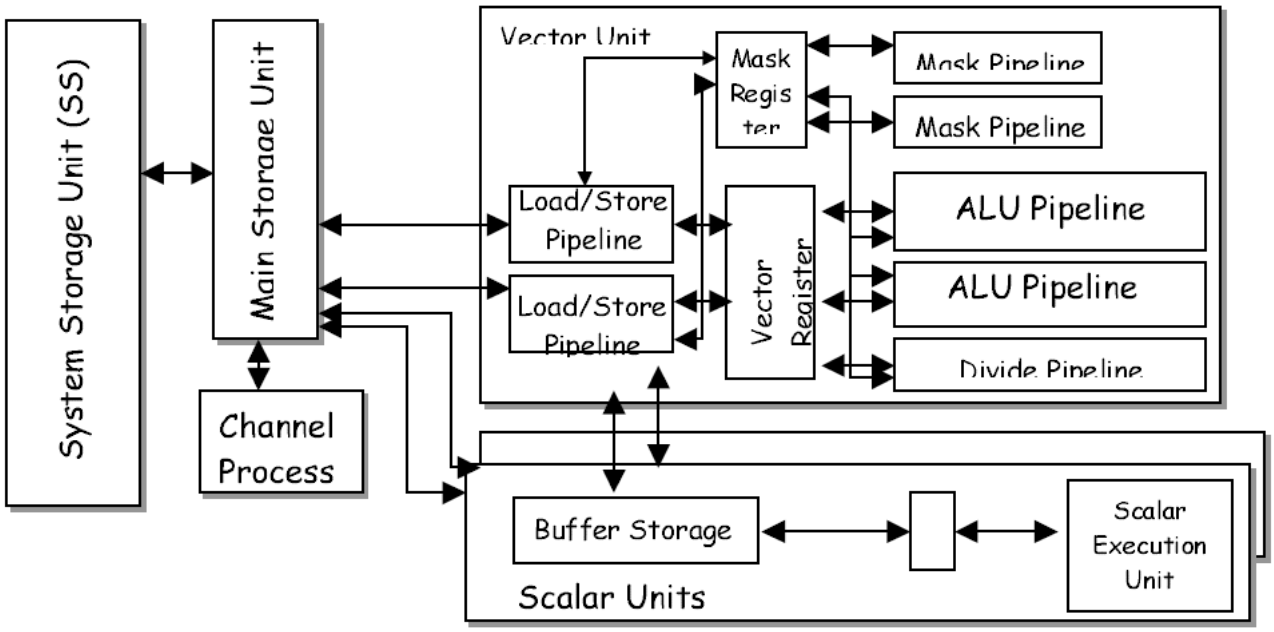
β. Αν υπήρχαν 2 θέσεις (slot) ανά μονάδα εκτέλεσης πως θα άλλαζε το παραπάνω;

γ. Με ποιο τρόπο μπορεί να μειωθεί ο χρόνος εκτέλεσης της ακολουθίας εντολών;

https://www.tutorialspoint.com/what-are-the-types-of-operands-fetch-policies

https://computerscience.chemeketa.edu/cs160Reader/ComputerArchitecture/MachineCycle.html

2. Consider the execution of the following compound vector function (CVF) for the vector processing unit below (it is equipped with two load/store pipelines plus five functional pipelines):
A(I) = B(I) * C(I) + D(I) * E(I) + F(I) * G(I)
for I=1, 2, ..., N. Initially, all vector operands are in memory, and the final vector result must be store in memory. Show the space-time diagram, for pipelined execution of the CVF. Note that, two vector loads can be carried out simultaneously on the two vector access pipes. At the end of computation, one of the two access pipes is used for storing the A array.



Για τον διανυσματικό επεξεργαστή του σχήματος δώστε το διάγραμμα χώρου-χρόνου για την πράξη A(I) = B(I) * C(I) + D(I) * E(I) + F(I) * G(I), I=1, 2, ..., N επεξηγώντας τις εγγραφές. Αρχικά όλα τα διανύσματα βρίσκονται στη μνήμη. Το αποτέλεσμα πρέπει να εγγραφεί στη μνήμη.

3. In this question we will analyze the performance of the following C program on a multithreaded architecture. You should assume that arrays A, B and C do not overlap in memory.

C code

```
for (i=0; i<328; i++) {
    A[i] = A[i] * B[i];
    C[i] = C[i] + A[i];
}
```

Our machine is a single-issue, in-order processor. It switches to a different thread every cycle using fixed round robin scheduling. Each of the N threads executes one instruction every N cycles. We allocate the code to the threads such that every thread executes every Nth iteration of the original C code.
Integer instructions take 1 cycle to execute, floating point instructions take 4 cycles and memory instructions take 3 cycles. All execution units are fully pipelined. If an instruction cannot issue because its data is not yet available, it inserts a bubble into the pipeline, and retries after N cycles.
Below is our program in assembly code for this machine for a single thread executing the entire loop.

```
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
```

a. We allocate the assembly code of the loop to N threads such that every thread executes every Nth iteration of the original loop. Write the assembly code that one of the N threads would execute on this multithreaded machine.
b. What is the minimum number of threads this machine needs to remain fully utilized issuing an instruction every cycle for our program?
c. Could we reach peak performance running this program using fewer threads by rearranging the instructions? Explain briefly.
d. What will be the peak performance in flops/cycle for this program?

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 Ν εκτελούν μία

C code

```
for (i=0; i<328; i++) {
    A[i] = A[i] * B[i];
    C[i] = C[i] + A[i];
}
```

εντολή κάθε Ν κύκλους. Κατακερματίζουμε τον κώδικα σε νήματα έτσι ώστε κάθε νήμα να εκτελεί κάθε Νη επανάληψη του αρχικού προγράμματος C.
Οι εντολές ακέραιων χρειάζονται 1 κύκλο για να εκτελεστούν, οι εντολές κινητής

```
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
```

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 Ν κύκλους.
Παρακάτω είναι το πρόγραμμά μας σε κώδικα μηχανής για αυτό το μηχάνημα για ένα μόνο νήμα που εκτελεί ολόκληρο τον βρόχο.

α. Εκχωρούμε τον κώδικα του βρόχου σε Ν νήματα έτσι ώστε κάθε νήμα εκτελεί κάθε Νη επανάληψη του αρχικού βρόχου. Γράψτε τον κώδικα μηχανής που θα εκτελούσε ένα από τα Ν 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;