Οργάνωση και Σχεδίαση Υπολογιστών (ECE 219)
Χειμερινό Εξάμηνο 2021-2022

Εργαστήριο 7

1. Επέκταση της μικρο-αρχιτεκτονικής διοχέτευσης του MIPS

Στο εργαστήριο αυτό θα πρέπει να κάνετε επεκτάσεις στην αρχιτεκτονική του MIPS που ολοκληρώσατε στο Εργαστήριο 6.

Υλοποίηση εντολών αλλαγής ροής προγράμματος (8 μονάδες)

Να επεκταθεί η προηγούμενη μικρο-αρχιτεκτονική όπως αναπαρίσταται και στην εικόνα 1) ώστε να μπορεί να εκτελέσει και τις παρακάτω εντολές διακλάδωσης MIPS:

- **`j Label`**
- **`beq rs, rt, Label`**
- **`bne rs, rt, Label`**

Εκτός από την σωστή εκτέλεση των εντολών αυτών, θα πρέπει να υλοποιήσετε και οποιαδήποτε επέκταση μηχανισμών προώθησης (forwarding), καθυστέρησης (stalling) και εκκένωσης (flushing) που είναι δυνατόν να προκαλέσουν οι επιπλέον αυτές εντολές. Για παράδειγμα, σκεφτείτε την επίδραση που έχουν οι εντολές διακλάδωσης στις εντολές που είναι ήδη μέσα στο pipeline. Η εκτέλεση της εντολής διακλάδωσης θα πρέπει να γίνεται στο στάδιο EX (και το αποτέλεσμα να προωθείται από το στάδιο MEM, όπως είδαμε και στο μάθημα).

Θα πρέπει να χρησιμοποιήσετε και να επεκτείνετε το αρχείο *control.v* και το αρχείο *cpu.v* που αναπτύξατε στο προηγούμενο εργαστήριο. Ο κώδικας σας θα πρέπει να τρέχει σωστά για κάθε περίπτωση όπως για παράδειγμα για τον κώδικα assembly που σας δίνεται στο *testbench.v* (το οποίο και δεν θα πρέπει να αλλάξετε). Στα σχόλια του *testbench.v* μπορείτε να βρείτε και τις αναμενόμενες τιμές των καταχωρητών και της μνήμης μετά από κάθε εκτελούμενη εντολή και στις δύο επαναλήψεις του loop. Θεωρούμε ότι κάθε καταχωρητής αρχικοποιείται με την τιμή `reg[i] = i`.

Η εξέτασή σας θα επικεντρωθεί στην σωστή λειτουργικότητα των εντολών διακλάδωσης.

Σημαντική Επισήμανση: η σωστή λειτουργικότητα του testbench απαιτεί να ονομάσετε κάποια συγκεκριμένα σήματα στην CPU *(cpu.v)* ως εξής:

(a) το σήμα ελέγχου του πολυπλέκτη επιλογής του PC μεταξύ του PC+4 και της διεύθυνσης άλματος του branch, θα πρέπει να έχει το όνομα *PCSrc*.

(b) το σήμα ελέγχου που δημιουργείται από το control unit *(control.v)* και είναι 1 όταν πρέπει να δημιουργηθεί μία φυσαλίδα (NOP) στο στάδιο EX λόγω stall ή flush, θα πρέπει να έχει το όνομα *bubble_idex*.

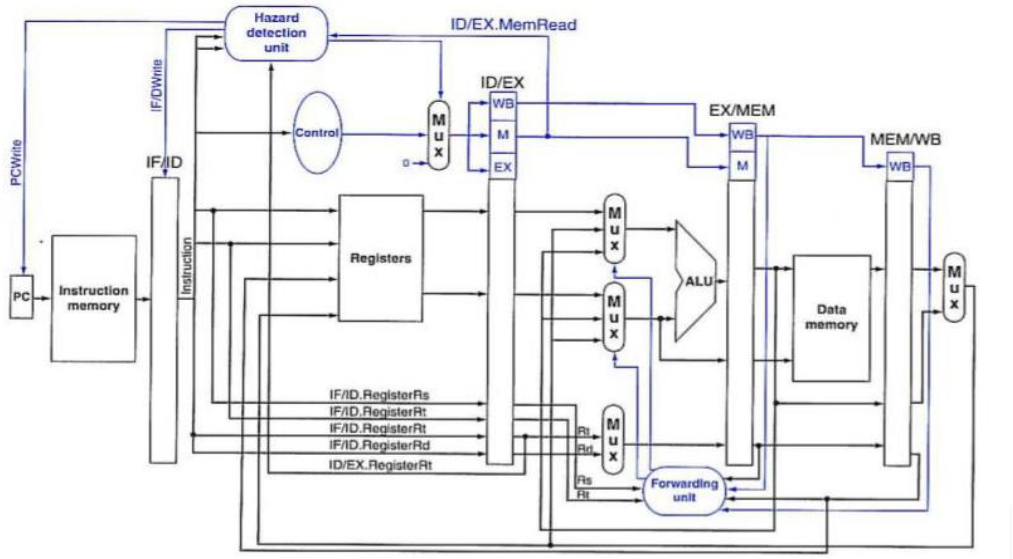

Figure 1. Αρχιτεκτονική MIPS με μηχανισμό διοχέτευσης (pipeline), μονάδα ανίχνευσης κινδύνου και μονάδα προώθησης (bypass). Η αρχιτεκτονική αυτή μπορεί να εκτελέσει εντολές format-R, καθώς και εντολές φόρτωσης, αποθήκευσης (load/store). Πάνω σε αυτήν την αρχιτεκτονική θα χτίσετε το κύκλωμα του Εργ. 7.

Αυτοί οι περιορισμοί γίνονται για να γίνει σωστά compile και simulate ο κώδικας του testbench:

```
string_manipulation pipe0(clock, reset, cpu0.PCSrc, cpu0.bubble_idx,
cpu0.instr, cpu0.IFID_instr, stringvar);
```

Αρχείο Καταχωρητών (Register File) (3 μονάδες)

Μέχρι τώρα, έχουμε υποθέσει ότι οι καταχωρητές του MIPS γράφονται στην αρνητική ακμή του ρολογιού. Αυτή η υπόθεση μας βοηθάει στο να μειώσουμε τον αριθμό των εξαρτημένων εντολών που διαβάζουν έναν καταχωρητή που γράφεται από μία προηγούμενη εντολή. Σε πραγματικές όμως μικρο-αρχιτεκτονικές, οι καταχωρητές διαβάζονται και γράφονται στην θετική ακμή του ρολογιού όπως όλοι οι υπόλοιποι καταχωρητές.

Το υποερώτημα αυτό σας ζητάει να πραγματοποιήσετε ότι αλλαγές απαιτούνται στο αρχείο καταχωρητών ώστε αυτοί και να διαβάζονται και να γράφονται στην θετική ακμή του ρολογιού.

Να προσέξετε ιδιαίτερα την περίπτωση που στον ίδιο κύκλο μηχανής μία εντολή γράφει σε έναν καταχωρητή από τον οποίο διαβάζει μία άλλη εντολή. Ενώ η εγγραφή στην αρνητική ακμή του ρολογιού αντιμετώπιζε αυτό το πρόβλημα στις προηγούμενες υλοποιήσεις, σε αυτήν την υλοποίηση θα πρέπει εσείς να επιλύσετε αυτό το πρόβλημα και να εξασφαλίσετε ότι τα δεδομένα που διαβάζονται είναι τα σωστά. Επίσης η ανάγνωση των καταχωρητών θα πρέπει να είναι σύγχρονη με την θετική ακμή του ρολογιού.

Η νέα μικρο-αρχιτεκτονική θα πρέπει να εκτελεί σωστά τον κώδικα της προηγούμενης ερώτησης.