ΟΡΓΑΝΩΣΗ ΥΠΟΛΟΓΙΣΤΩΝ

ΑΝΑΦΟΡΑ ΕΡΓΑΣΙΑΣ 2

Ομάδα: LAB31243955

Θεόδωρος Κουτσαβδής 2017030145

Ιωάννης Οικονομόπουλος 2017030048

4η Φάση εργασίας "Δημιουργία επεξεργαστή πολλαπλών κύκλων"

ΠΕΡΙΓΡΑΦΗ

Στη συγκεκριμένη φάση εργασίας σκοπός ήταν η επέκταση και η μετατροπή της 1ης εργασίας (επεξεργαστής ενός κύκλου), προκειμένου να δημιουργηθεί ο επεξεργαστής πολλαπλών κύκλων. Για την επίτευξη του συγκεκριμένου σκοπού απαιτήθηκαν αρκετές αλλαγές τόσο στο datapath όσο και στο control μέρος της υλοποίησης.

Ειδικότερα, όσον αφορά **datapath** μέρος, οι αλλαγές που έγιναν είναι οι εξής:

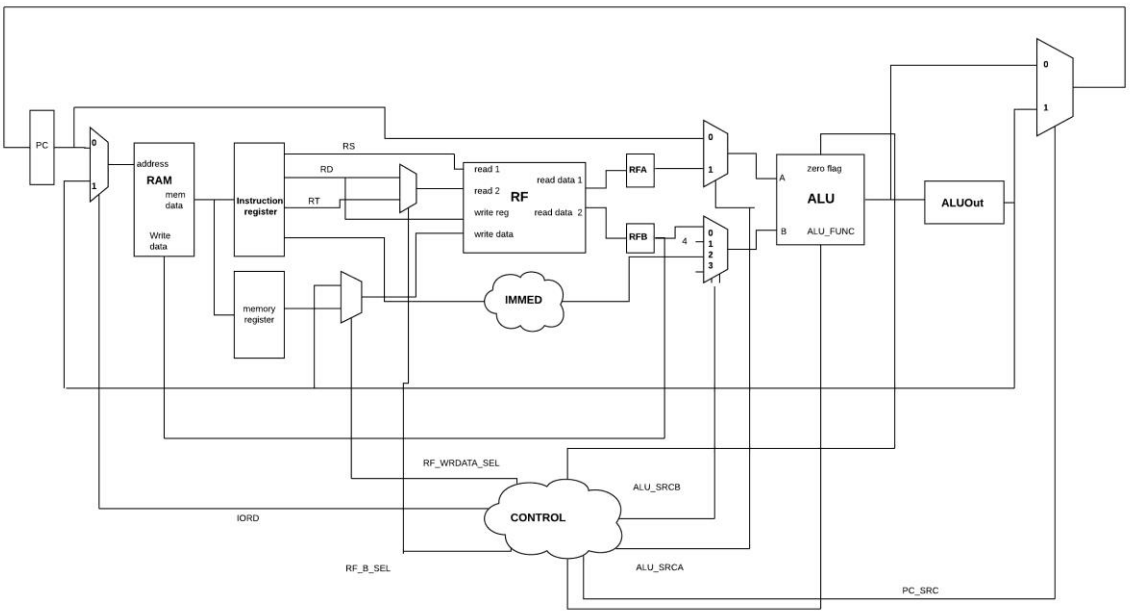
- Δημιουργήθηκαν νέοι καταχωρητές οι οποίοι τοποθετήθηκαν ανάμεσα σε κάθε stage. Συγκεκριμένα, οι νέοι καταχωρητές είναι: Instruction_register, MemData_Register, RFA_Register, RFB_Register και ALUOut_Register.
- Αφαιρέθηκαν οι adders από τον καταχωρητή PC στο μέρος IFSTAGE. Πλέον, η μετακίνηση του PC είτε κατά +4 είτε κατά Immed γίνεται μέσω της ALU (EXSTAGE).
- Στο μέρος EXSTAGE, προστέθηκε ένας ακόμη πολυπλέκτης ανάμεσα στο RFA και την είσοδο της ALU **,** ο οποίος παίρνει ως εισόδους είτε την έξοδο του καταχωρητή RFA είτε τον PC, ενώ ο ήδη υπάρχων πολυπλέκτης ανάμεσα στο RFB και την άλλη είσοδο της ALU, μετατράπηκε από 2 προς 1 σε 4 προς 1, αφού παίρνει πλέον ως εισόδους το RFB,το Immed και το +4.

Όσον αφορά το **control** μέρος, ήταν δεδομένο ότι θα χρειαζόντουσαν αλλαγές για την υλοποίηση του επεξεργαστή πολλαπλών κύκλων. Για το λόγο αυτό, το control μέρος υλοποιήθηκε, δημιουργώντας μια μηχανή πεπερασμένων καταστάσεων FSM. Ο λόγος για τον οποίο έγινε κάτι τέτοιο είναι ότι σε κάθε state είναι απαραίτητη η χρησιμοποίηση διαφορετικών σημάτων ελέγχου, αφού εκτός των εντολών που πρέπει να πραγματοποιηθούν είναι απαραίτητος και ο συγχρονισμός και η αύξηση του καταχωρητή PC μέσω της ALU. Με τον τρόπο αυτό, παρατηρείται ότι το κάθε state κρατάει πολύ λιγότερο χρόνο και ότι το σύνολο των state υλοποιούν την κάθε εντολή.

Οι παραπάνω αλλαγές πραγματοποιήθηκαν έτσι ώστε να δημιουργηθεί ένας πιο γρήγορος επεξεργαστής (multicycle) σε σχέση με το αντίστοιχο singlecycle. Κάτι τέτοιο επιβεβαιώθηκε στην πράξη, αφού ο επεξεργαστής ενός κύκλου πραγματοποιούσε όλες τις εντολές ανάλογα με τον κύκλο της πιο αργής εντολής, ενώ ο αντίστοιχος επεξεργαστής πολλαπλών κύκλων παίρνει εκτελεί τις εντολές ακριβώς στο χρόνο που χρειάζεται η κάθε μια ξεχωριστά.

ΣΧΕΔΙΑΣΜΟΣ

Οι αλλαγές που αναφέρθηκαν παραπάνω απεικονίζονται παραστατικά από το παρακάτω σχήμα, το οποίο αποτελεί τη σχεδίαση του επεξεργαστή που υλοποιήθηκε:



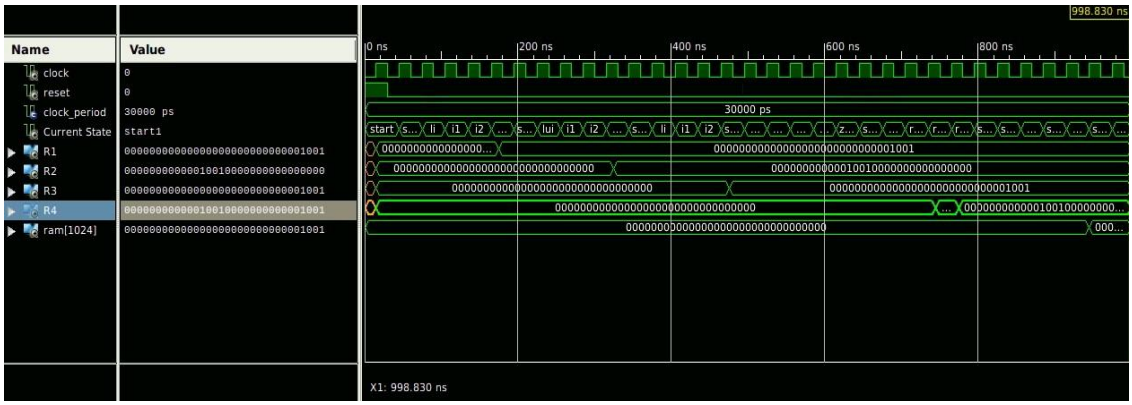
ΕΠΕΞΕΡΓΑΣΊΑ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΑΝΑΦΟΡΑΣ 1

ΠΡΟΓΡΑΜΜΑ ΑΝΑΦΟΡΑΣ 2

ΠΡΟΓΡΑΜΜΑ ΑΝΑΦΟΡΑΣ 3

Περιλαμβάνονται όλες οι υπόλοιπες εντολές που δεν υπήρχαν στο πρόγραμμα αναφοράς 1 και 2. Συγκεκριμένα, το πρόγραμμα εκτελεί:



Το συγκεκριμένο testbench επαληθεύει το σύνολο των εντολών:

```
Li     $1,   9
Lui    $2,   9
Li      $3,   9
Beq    $3,   $1,   2
-----------------------
-----------------------
Add    $4,   $1,   r2
Sb      $4,   0($zero)
```

5η Φάση εργασίας “Δημιουργία pipeline επεξεργαστή”

ΠΕΡΙΓΡΑΦΗ

Στη συγκεκριμένη φάση εργασίας σκοπός ήταν ο σχεδιασμός ενός pipeline επεξεργαστή. Για την δημιουργία του συγκεκριμένου επεξεργαστή απαιτήθηκαν αλλαγές τόσο στο datapath όσο και στο control μέρος της υλοποίησης.

Ειδικότερα, όσον αφορά **datapath** μέρος, οι αλλαγές που έγιναν είναι οι εξής:

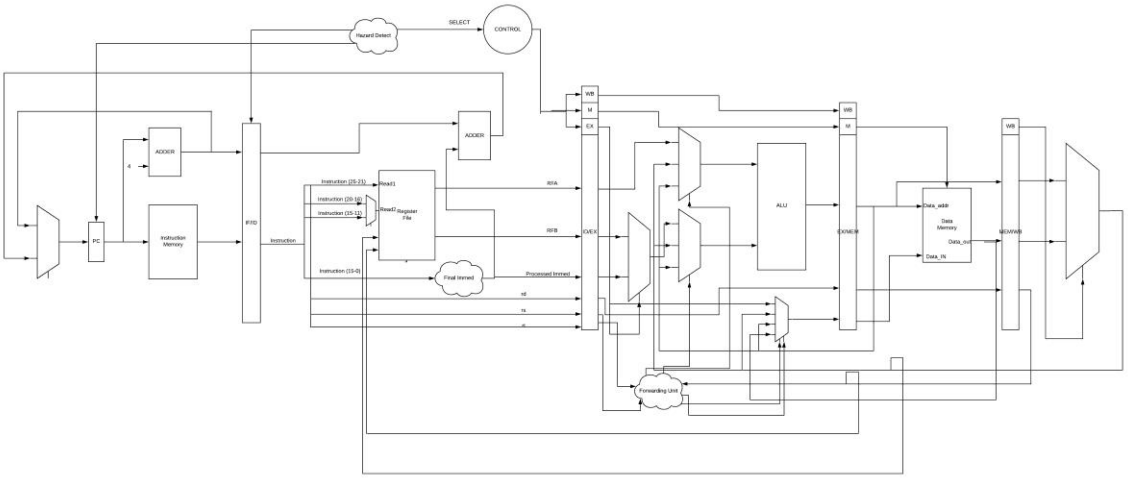
- Δημιουργήθηκαν νέοι pipeline καταχωρητές οι οποίοι τοποθετήθηκαν ανάμεσα σε κάθε stage. Για το λόγο αυτό δημιουργήθηκαν νέα modules: (IF/ID), (ID/EX), (EX/MEM), (MEM/WB). Συγκεκριμένα, το **(IF/ID)** έχει τον instruction register και τον PC, το **(ID/EX)** έχει τους registers RFA, RFB,RS,RT,Rd,Immed ,OpCode,OpCode1,EX, M, WB, το **(EX/MEM)** έχει τους registers ALU_Out,MEM_DataIn, OpCode1, RD, M, W και το **(MEM/WB)** έχει τους registers MEM_DataOut, ALU_Out, RD, WB.
- Για το μέρος EXSTAGE δημιουργήθηκαν 4 πολυπλέκτες. Συγκεκριμένα, ένας 2 προς 1, που επιλέγει ανάμεσα στο RFB και το Immed, και τρεις 4 προς 1 για την υλοποίηση του forward(hazards).
- Στο IFSTAGE επέστρεψε ο ένας adder για τον καταχωρητή PC, ο οποίος τον αυξάνει κατά +4.
- Στο DECSTAGE προστέθηκε ένας adder, οποίος προσθέτει τον καταχωρητή PC με το Immed. Η υλοποίηση αυτή έγινε στο συγκεκριμένο βήμα για την αντιμετώπιση Control Hazards, αν προκύψει τύπου branch εντολή.
- Προστέθηκε επίσης το Forward_unit στο οποίο με κατάλληλους ελέγχους επιτυγχάνεται η προώθηση σημάτων για την αντιμετώπιση των Data Hazards.
- Τέλος, υλοποιήθηκε το Detection_Unit το οποίο με τους κατάλληλους ελέγχους αντιμετωπίζει Data Hazards χρησιμοποιώντας bubbles (STALL).

Όσον αφορά το **control** μέρος της υλοποίησης πραγματοποιήθηκαν αλλαγές προκειμένου να δημιουργηθεί ο pipeline επεξεργαστής. Συγκεκριμένα, τα σήματα ελέγχου του control περνούν μέσα από τους καταχωρητές EX, M, WB των βαθμίδων ID/EX, EX/MEM, MEM/WB και χρησιμοποιούνται από στο STAGE που είναι απαραίτητο. Έτσι, συμπεραίνεται το γεγονός ότι το control δεν πρέπει να λειτουργεί ως μηχανή πεπερασμένων καταστάσεων FSM, και γι αυτό το λόγο τα σήματα δίνονται όλα μαζί, σύμφωνα με το OpCode, στη βαθμίδα DECSTAGE και μεταφέρονται μέσω των καταχωρητών στις άλλες βαθμίδες.

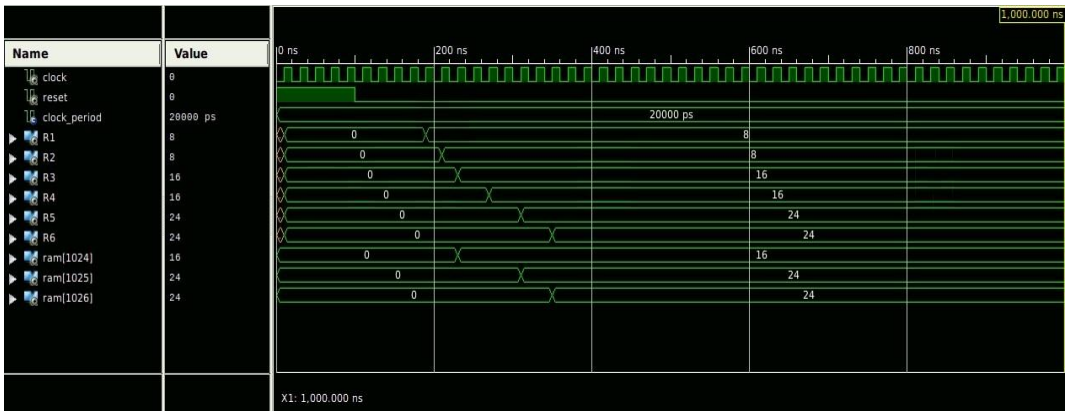
Όλες αυτές οι αλλαγές σε συνδυασμό με την ορθή αντιμετώπιση των Hazards είχαν ως αποτέλεσμα τη δημιουργία ενός pipeline επεξεργαστή, ο οποίος όπως και αναμενόταν ήταν αρκετά γρηγορότερος και από τον singlecycle αλλά και από τον multicycle επεξεργαστή.

ΣΧΕΔΙΑΣΜΟΣ

Οι αλλαγές που αναφέρθηκαν παραπάνω απεικονίζονται παραστατικά από το παρακάτω σχήμα, το οποίο αποτελεί τη σχεδίαση του επεξεργαστή που υλοποιήθηκε:



ΕΠΕΞΕΡΓΑΣΊΑ ΑΠΟΤΕΛΕΣΜΑΤΩΝ



Το συγκεκριμένο testbench επαληθεύει το σύνολο των εντολών:

```
Li      $1,     8                   --  $1 = 8
Add     $2,     $1,     ($zero)     -- forward /  $2 = 8
Add     $3,     $2,     $1          -- double forward /  $3 = 16
Sw      $3,     ($zero)             -- Store forward / ram[1024] = 16
Lw      $4,     ($zero)             -- $4 = 16
Add     $5,     $4,     $1          --  Stall /forward / $5 = 24
Sw      $5,     4($zero)            --  Store forward / ram[1025] = 24
```

```
Lw      $6,     4($zero)        --  $6 = 24
Sw      $6,     8($zero)        -- (tricky) forward from memstage / ram[1026] = 24
```