Οργάνωση Υπολογιστών

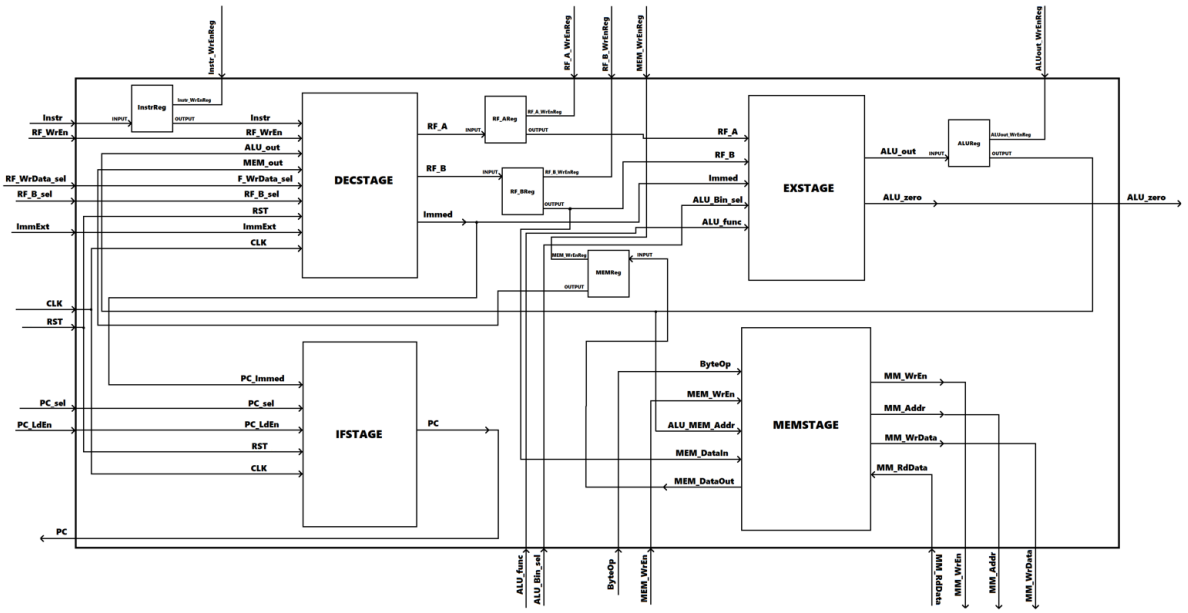
Εργασία 2

Παπαδόπουλος Αλέξανδρος 2014030071

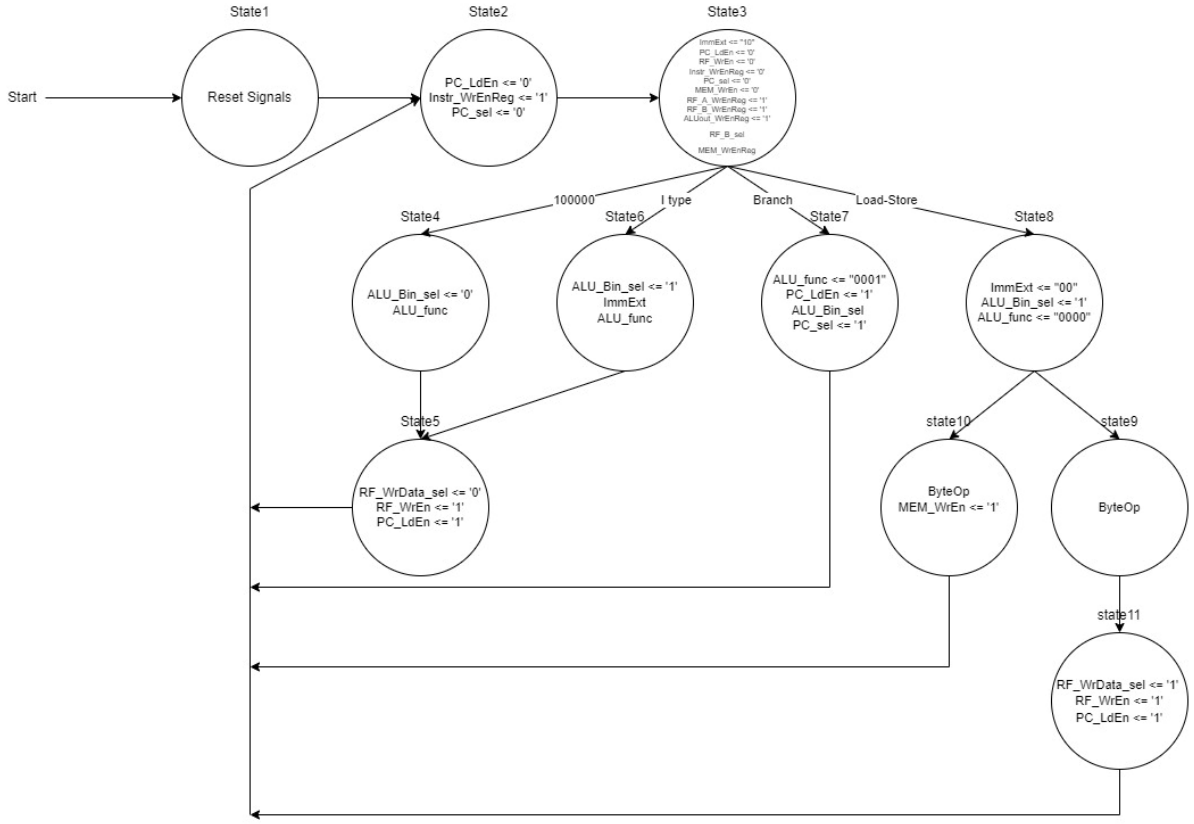
Σε αυτήν την εργασία μετατρέψαμε τον επεξεργαστή από single cycle σε multi cycle. Η μετατροπή έγινε με καταχωρητές μεταξύ των stages, με αυτό τον τρόπο αποθηκεύονται οι τιμές μετά από κάθε stage και έτσι μπορούμε να έχουμε πολλαπλούς κύκλους για μία εντολή.

Αξιοποιώντας τον επεξεργαστή single cycle που σχεδίασα στην πρώτη εργασία σχεδίασα το datapath και το control του multi cycle processor. Πιο συγκεκριμένα, στο datapath πρόσθεσα μεταξύ κάποιων βαθμίδων registers για να αποθηκεύονται οι τιμές για τον επόμενο κύκλο. Αυτό έχει σαν αποτέλεσμα η διάρκεια κάθε κύκλου να μικραίνει σημαντικά και έτσι μία εντολή να εκτελείται γρηγορότερα σε περισσότερους κύκλους. Σύνδεσα έναν καταχωρητή με το instruction και με το decstage, έναν με το RF_A και RF_B, έναν με την έξοδο της ALU και τέλος έναν με την έξοδο MEM, η υπόλοιπη σχεδίαση του datapath δεν έχει αλλαγές από εκείνο του single cycle. Για το control σχεδίασα μία FSM με διαφορετικά states τα οποία αλλάζουν ανάλογα με το τωρινό state και την είσοδο, οπότε είναι μία Mealy FSM. Το πλεονέκτημα που προσφέρει μία Mealy FSM σε σύγκριση με μία Moore FSM είναι ότι μειώνεται σε μεγάλο βαθμό ο αριθμός των state της μηχανής μας. Έτσι γίνεται ευκολότερος ο έλεγχος της ορθότητας του κώδικα μας. Στη συνέχεια σύνδεσα control και datapath σε ένα μεγαλύτερο Module το PROCESSOR_MC και σε ένα testbest το module αυτό με τη ram. Στο πρώτο stage γίνεται το fetch instruction από το IFSTAGE και υπολογίζεται η πράξη PC = PC + 4. Στο δεύτερο stage γίνεται το decode της εντολής από το DECSTAGE. Για το τρίτο stage έχουμε το execution της εντολής από το EXSTAGE. Στο τέτατρο stage γίνεται προσπέλαση στη μνήμη και αν είναι store ολοκληρώνεται η εντολή ενώ αν είναι load χρειάζεται ένα ακόμα stage.

DATAPATH



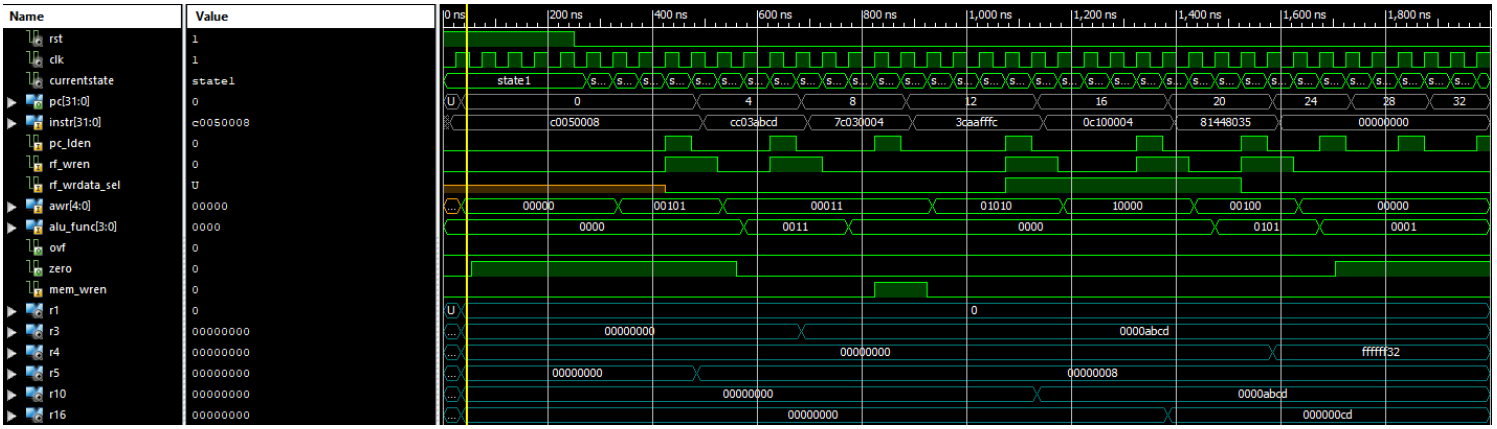
FSM



Αποτελέσματα Προγραμμάτων Αναφοράς

Πρόγραμμα αναφοράς 1:

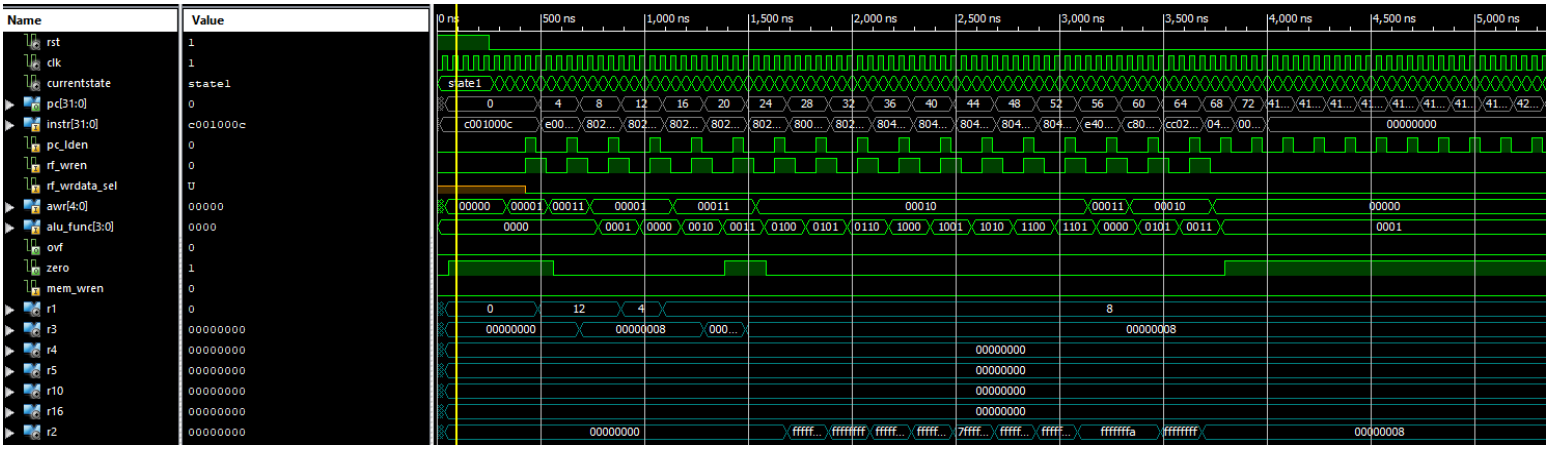
```
00: addi r5, r0, 8
04: ori  r3, r0, 0xABCD
08: sw   r3, 4(r0)        // γράφει στην διεύθυνση 0x4 => 0x404 την τιμή 0x0000ABCD
0C: lw  r10, -4(r5)       // διαβάζει από την διεύθυνση 0x4 => 0x404 την τιμή 0x0000ABCD
10: lb  r16, 4(r0)        // διαβάζει byte από την διεύθυνση 0x4 => 0x404 την τιμή 0x000000CD
14: nand r4, r10, r16
```



Πρόγραμμα αναφοράς με όλες τις εντολές ISA:

Οι εντολές εκτελούνται με την παρακάτω σειρά

addi r1, r0, 12
li r3, 8
sub r1, r1, r3
add r1, r1, r1
and r3, r1, r0
not r2, r1, r0
nand r2, r0, r0
nor r2, r1, r0
sra r2, r2
srl r2, r2
sll r2, r2
rol r2, r2
ror r2, r2
lui r3, 8
nandi r2, r0, 8
ori r2, r0, 8
bne r0, r0, 0xffff
beq r0, r0, 0xfff
b 0xff



Στη συγκεκριμένη εργασία καταλάβαμε ότι μπορούμε να κάνουμε τον επεξεργαστή μας γρηγορότερο χρησιμοποιώντας περισσότερους από έναν κύκλο για την εκτέλεση μίας εντολής το οποίο όμως αυξάνει τη συχνότητα του ρολογιού. Μάθαμε να υλοποιούμε FSM στη γλώσσα VHDL και εξομοιωθήκαμε περαιτέρω με αυτήν.

Σημείωση: Δεν υλοποιήθηκε το Pipeline.