Κατσούπης Ευάγγελος
2017030077

Σχεδίαση μονάδας αριθμητικών και λογικών πράξεων (ALU), και αρχείου καταχωρητών (Register File).

- Μονάδα αριθμητικών και λογικών πράξεων ALU. Το Arithmetic Logic Unit είναι ένα συνδιαστικό κύκλωμα το οποίο αποτελείτε από 3 εισόδους, τελεστέοι A και B και ελενκτής πράξης OpCode αλλά και από διάφορα σήματα εξόδου όπως το Outpout (αποτέλεσμα με καθυστέρηση $10ns$), ZeroFlag (σήμα ενεργοποιημένο για μηδενικό αποτέλεσμα) αλλά και τα OverFlowFlag, Cout σήματα που υποδηλώνουν υπερχείλιση και κρατούμενο εξόδου αντίστοιχα. Συγκεκριμένα οι πράξεις που πραγματοποιούντε, ανάλογα με το OpCode είναι:

OpCode	Out	OpCode	Out	OpCode	Out
0000	A + B	0001	A - B	0010	A and B
0011	A or B	0100	!A	0101	A nand B
0110	A nor B	1000	(int)A$\gg$1	1001	unsign(A) $\gg$ 1
1010	A$\ll$1	1100	rol A	1101	ror A

  Στις περιπτώσεις της προσθαφαίρησης μπορεί να ενεργοποιηθούν τα ZeroFlag και Cour.

  Για τον υπολογισμό των παραπάνω έγινε χρήση των επιπλέων βιβλιοθηκών: Numeric std και STD Logic Signed.

- Αρχείο καταχωρητών:
  Αρχικά δημιουρήθηκε το component του καταχωρητή το οποίο σε κάθε ακμή του ρολογιού έχει σαν outpout (32 bits) ότι και το input (32 bits) αρκέι το WE να είναι ενεργό. Στην περίπτωση του reset η έξοδος αρχικοποιήτε σε 0.
  Στην συνέχεια και με την χρήση του παραπάνω καταχωρητή, ορίστηκε το register file το οποίο περιέχει 32 τέτοιους καταχωριτές. Ποιο συγκεκριμένα δέχεται σαν είσοδο 3 5-bit σήματα, διευθήσεις που σηματοδοτούν ποιοι καταχωριτές θα διαβαστούν και και (Dout1,2) και ποιος θα πάρει νέα τιμή (Awr). Αξίζει να σημειωθεί πώς για να πραγματοποιηθεί η εγκραφή θα πρέπει το WrEn να είναι ενεργό. Ο R0 έχει πάντα μηδενική τιμή.

Σχεδίαση των βασικών βαθμίδων του datapath ενός απλού επεξεργαστή.

- Υλοποίηση κύριας μνήμης $2048x3$:
  Ηλοποιήθηκε η κύρια μνήμη του επεξεργαστή ενός IPC σύμφωνα με το component που δόθηκε. Για την αρχηκοπιήση της μνήμης χρησιμοποιήθηκε το αρχείο rom.data το οποίο περιέχει το σύνολο των εντολών (text segment) που θα υλοποιηθούν, Δυναμικά δημιουργήτε το data segment που αποτελείτε από τα δεδομένα που θα αποθηκεύονται/διαβάζονται κατά την εκτέλεση του προγράμματος. Η κύρια μνήμη αλληλεπιδρά με το IF stage (διεύθηνση επόμενης εντολής, instraction για εκτέλεση), αλλά και με το Memory stage για την αποθήκευση/διαβάζμα δεδομένων.

- Βαθμίδα ανάκλησης εντολών:
  Κατά το instraction fetch ένας αθροιστής υπολογίζει την επόμενη τιμή του PC, προστέθοντάς την κατά 4 και ένας άλλος προστέθοντάς την κατά 4 αλλά και προστέθοντας το $Immedvalue$ που δίνεται. Η επιλογή της επόμενης διεύθηνσης καθορίζετε από το σήμα PC sel το οποίο ορίζει την τιμή εξόδου του πολυπλέκτη που κρατά τις προηγούμενες τιμές. Βέβαια ο για να πάρουμε την νέα διέυθηνσφ θα πρέπει το $PCLdEn$ να είναι ενεργό. Για τον έλενχο της σωστής λειτουργείας το ενώσαμε με την βαθμίδα της κύριας μνήμης.

- Βαθμίδα αποκωδικοποίησης εντολών:
  Σαν είσοδος ορίζεται το instraction το οποίο διαβάστικε από την μνήμη και ποιο συγκεκριμένα κομμάτια αυτού ορίζουν την τα σήματα εισόδου του RF που είναι συνδεδεμένο ($bits25-21 \rightarrow RF_A$, $15-11()20-16 \rightarrow RF_B$). Το σήμα RF B sel ελένχει ποιος θα είναι ο δεύτερος καταχορητής που θα διαβαστεί (εξαρτάτε από την εντολή).Τα δεδομένα για εγγραφή, όταν είναι ενεργό το RF WrEn, θα προέρχονται είτε από την μνήμη είτε από την ALU. Για την διαχήρηση του Immediate δημιουργήθηκε το component cloud unit το οποιό, σύμφωνα με το σήμα ελέγχου ImmExt, το οποίο προέρχεται από το control, μετατρέπει το Immediate του instraction σε $32bits$ κάνοντας ένα από τα παρακάτω:

ImmExt	Out
00	Zero fill
01	Sign Extend
10	Zero Fill & Shift (immed to 31-16)
11	Shift by 2 and sign extend

- Βαθμίδα εκτέλεσης εντολών:
  Σε αυτό το στάδιο συνδέσαμε την ALU με έναν πολυπλέκτη. Σαν πρώτος τελεστέος της ALU είναι πάντα ο $RF[rs]$ (RF A of Decode stage) Την έξοδο του πολυπλέκτη, δεύτερος τελεστέος της ALU την καθορίζει το σήμα του Control, $ALUBinSel$ που επιλέγει είτε μια Immed τιμή είτε έναν καταχωριτή.

- Βαθμίδα πρόσβασης μνήμης:
  Σε αυτή την κατάσταση ελένχοντε και επεξεργάζονται τα σήματα τα οποία θα πηγαίνουν στην κύρια μνήμη. Πιο συγκεκριμένα σύμφωνα με με το σήμα ελέχου $ByteOp$ ελέγχεται η μορφή του σήματος σε περίπτωση εγγραφής/ανάγνωσης.Στην περίπτωση του lb τότε στο σήμα που φτάνει από την μνήμη πρέπει να εφαρμοστεί μάσκα για να αποθηκεύονται μόνο τα 8 LSB. Το ίδιο συμβαίνει και στην περίπτωση εγγραφής στην μνήμη.Τέλος για τις εντολές που επεξεργάζονται λέξεις πέρνουμε τα σήματα αυτούσια.
  Κατά την εκτέλεση ενώς προγράμματος υπάρχει πιθανότητα να γίνουν $store$ κάπια δεδομένα στο $DataSegment$ της RAM. Γιὰυτό τον λόγο προστίθενται στο σήμα $ALUMEMAddr$ η τιμή 1024 ($0x400$ στα $bits$ 12 - 2) για να αποθηκευθούν τα δεδομένα στα, δυναμικά δημιουργημένο $DataSegment$. Τα σήματα $MemWrEn$, $MemDataIn$ περνάνε στη μνήμη και το $MMRdData$ έιναι έξοδος του ΜΕΜΣΤΑΓΕ.

Ολοκλήρωση του datapath, δημιουργία του control, και σύνδεση datapath-control.

Για την δημιουργία του datapath ενώθηκαν όλα τα επιμέρους στάδια αφήνοντας μόνατα σήματα ελένγου. Σαν επόμενο βήμα εγκαθίστανται και το μοναδικό component της RAM Το οποίο ενώνεται με με το $IFstage$ αλλά και με το $MEMstage$.

Με βάση τα προγράμματα αναφοράς που δίνονται, δημιουργήθηκαν τα $rom.data$ αρχεία.
Στην συνέχεια ολοκληρώθηκε το $control$ το οποίο σύμφωνα με το $OpCode$, $Func$ και το σήμα ΑΛΥ Zero παράγει τα σωστά σήματα ελέγχου (Σύμφωνα πάντα με τα CHARIS).

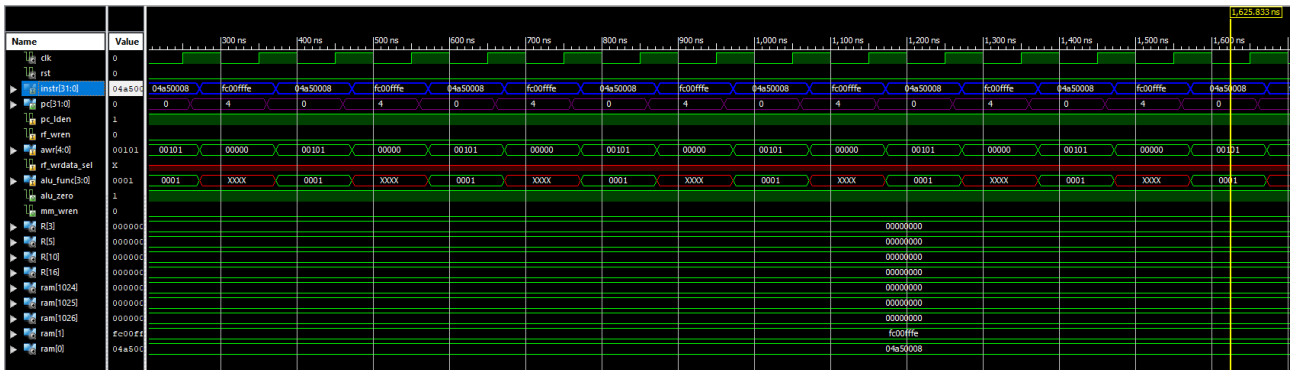
Τέλος, για την δημιουργεία του επεξεργαστή με $IPC = 1$ ενώθηκαν τα $DATAPATH, CONTROL$ και ένα μοναδικό $instance$ κύριας μνήμης ολοκληρώνοντας το $toplevel\ \ module$.

Προγράμματα αναφοράς

1. (a) addi r5,r0,8;
   (b) ori r3,r0,ABCD;
   (c) sw r3,4(r0);
   (d) lw r10,-4(r5);
   (e) lb r16,4(r0);
   (f) nand r4,r10,r16;

Παρατηρούμε πώς αρχικά ο $R5$ πέρνει την τιμή 8, με την επόμενη εντολή ο $R3$ παίρνει την τιμή $0xABCD$ $(R0 => 0)$ και με την τρίτη εντολή αποθήκευεται στην θέση 1025 του $R3$. Σε επόμενο κύκλο φορτώνεται στον $R10$ τα περιεχόμενα της προηγούμενη διεύθηνσης που κρατά ο $R5$ $(8 - 4 = 4 -> 1025)$ δηλαδή η τιμή $0x0ABCD$. Στην συνέχεια ο $R16$ παίρνει την τιμή $0x000000CD$ σαν αποτέλεσμα της lb. Τέλος ο $R4$ παίρνει το αποτέλεσμα του λογικού $nand$ των $R10$ και $R16$.

2. (a) bne r5,r5,8
   (b) b -2
   (c) addi r1,r0,1



Μπορούμε έυκολα να διακρίνουμε την αποτυχημένη διακλάδωση μιας και ο PC εναλλασσεται μεταξύ του 0 και του 4.
Δεδομένου αυτού η εντολή *addi* εκτελείτε ποτέ.