ΟΡΓΑΝΩΣΗ ΥΠΟΛΟΓΙΣΤΩΝ
ΑΝΑΦΟΡΑ ΑΣΚΗΣΗΣ #1

SINGLE CYCLE MIPS PROCESSOR

ΧΡΗΣΤΟΣ ΧΡΗΣΤΟΥ 2018030126
ΓΕΩΡΓΙΟΣ ΝΤΑΛΙΑΝΗΣ 2020030045

ΜΕΡΟΣ Α - ΕΙΣΑΓΩΓΗ:

Σκοπός της εργαστηριακής άσκησης πρώτα είναι η μελέτη και η κατανόηση της λειτουργίας του επεξεργαστή ενός κύκλου και έπειτα η υλοποίηση του σε VHDL. Η αρχιτεκτονική (Instruction Set) που χρησιμοποιήθηκε είναι μια παραλλαγή της αρχιτεκτονικής MIPS η οποία περιέχει τις πολύ βασικές εντολές (arithmetic, memory access, branches). Λόγω της μεγάλης πολυπλοκότητας του προβλήματος, η υλοποίηση έγινε με τη μέθοδο “Διαίρει και Βασίλευε”. Το τελικό πρόβλημα απλοποιήθηκε σε πολλά μικρότερα και απλούστερα. Στη συνέχεια υλοποιήθηκαν αυτά τα υποπροβλήματα κατασκευάζοντας τα απαραίτητα modules. Έπειτα modules συνδέθηκαν μεταξύ τους και δημιουργώντας μεγαλύτερα έως ότου δημιουργηθεί ο τελικός επεξεργαστής. Η υλοποίηση χωρίστηκε σε 3 μεγάλες φάσεις οι οποίες περιγράφονται αναλυτικά παρακάτω.

ΜΕΡΟΣ Β - ΥΛΟΠΟΙΗΣΗ:

ΠΡΩΤΗ ΦΑΣΗ:

Σε αυτή τη φάση σχεδιάστηκαν μια απλή Αριθμητική και Λογική Μονάδα (ALU) και το αρχείο καταχωρητών (Register File). Η ALU μπορεί να πραγματοποιήσει όλες τις πράξεις που περιγράφονται στο CHAR-Instruction-Set. Το Register File περιέχει 32 registers, έναν decoder 5:32 bits και δύο multiplexers 32:1 bits με 32-bit η κάθε είσοδος. Εδώ πρέπει να επισημανθεί ο τρόπος με τον οποίο υλοποιήθηκε ο decoder. Παρατηρήθηκε ότι η έξοδος του ισοδυναμεί με την πράξη bit shift left 1 << input.

```
constant one: std_logic_vector(31 downto 0) := x"00000001";
begin
    output <= std_logic_vector(unsigned(one) sll to_integer(unsigned(input))) after
latency;
end decoder;
```

Οι multiplexers φτιάχτηκαν ορίζοντας logic vector array. Αυτό ορίστηκε σε ένα εξωτερικό package (βλ. utils.vhd) προκειμένου να μπορεί να χρησιμοποιηθεί από πολλά modules. Το input στον κώδικα παρακάτω είναι array από 32 bit logic vectors.

```
architecture Structural of mux_32 is
begin
    output <= input(to_integer(unsigned(sel))) after
latency;
end Structural;
```

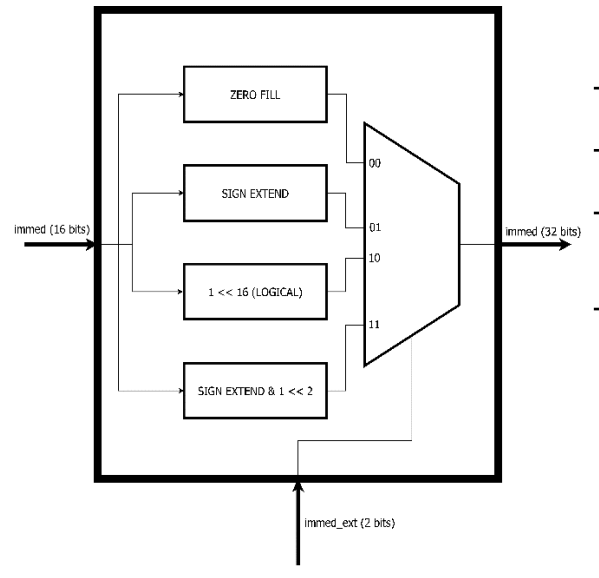
Το Register File έχει δυνατότητα σύγχρονης εγγραφής σε 1 καταχωρητή και ασύγχρονης ανάγνωσης 2 καταχωρητών. Ο καταχωρητής R0 έχει πάντα την τιμή 0x0000. Αυτό γίνεται έχοντας το write enable του πάντα στην τιμή ‘0’. Παρακάτω φαίνεται ο τρόπος με τον οποίο υλοποιήθηκε. Το write enable array είναι 32 bit logic vector με 1 bit για κάθε write enable του κάθε καταχωρητή. Τέλος στο Register File προστέθηκε και reset signal για αρχικοποίηση.

```
write_enable_array <= decoded_address_write and x"00000000" when write_enable = '0'
else decoded_address_write and x"fffffffe";
```

ΔΕΥΤΕΡΗ ΦΑΣΗ:

Σε αυτή τη φάση κατασκευάστηκαν τα 4 μεγάλα μέρη του DATAPATH. Αυτά είναι τα IFSTAGE, DECSTAGE, EXSTAGE MEMSTAGE. Το IFSTAGE η αλλιώς Instruction Fetch Stage είναι υπεύθυνο για την ανάρτηση εντολών προς εκτέλεση από τη μνήμη. Περιέχει τον καταχωρητή Program Counter (PC). Ωστόσο επειδή η μνήμη RAM μπορεί διευθυνσιτοδοτήσει μέχρι 2^{11} = 2048 θέσεις μνήμης η RAM δέχεται σαν είσοδο τα bits 12 έως και 2 του Program Counter, το οποίο ισοδυναμεί με την πράξη (PC >> 2) % 2^{11}.

Η αποκωδικοποίηση των εντολών γίνονται στο στάδιο DECSTAGE. Συγκεκριμένα σπάει η εντολή στα επιμέρους κομμάτια, δηλαδή στο rd, rt, rs και immediate και επιλέγονται οι επιμέρους καταχωρητές. Στον καταχωρητή rd γίνεται εγγραφή ενώ στους άλλους 2 ανάγνωση. Τέλος το immediate κομμάτι της εντολής από 16 bits μετατρέπεται σε 32 bits προκειμένου να μπορεί να χρησιμοποιηθεί από την ALU και το IFSTAGE. Η υλοποίηση του immediate extender φαίνεται παρακάτω.



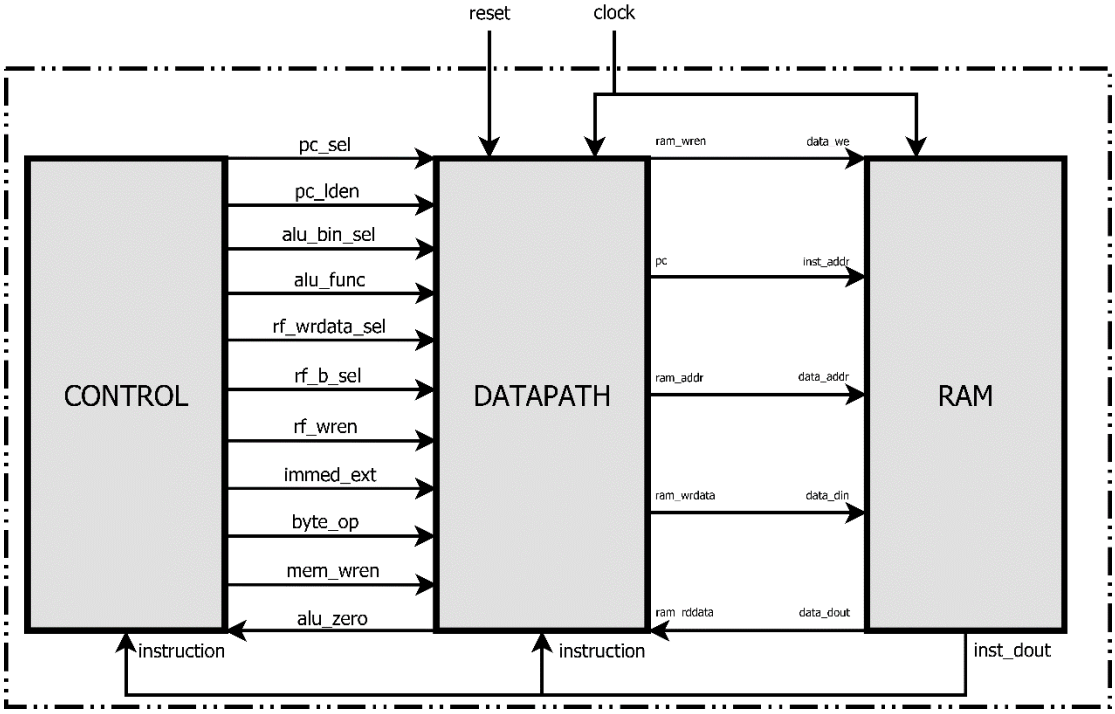
imm_ext (mux-select)	operation
00	zero fill
01	sign extend
10	<< 16 (logical)
11	sign extend and << 2

Η εκτέλεση των εντολών γίνεται στην βαθμίδα EXSTAGE (Execution Stage). Εδώ εκτελούνται όλες οι εντολές που έχουν να κάνουν με άμεση πράξη (add, sub κλπ.) αλλά και οι εντολές που πραγματοποιούν έμμεση πράξη (beq, sw κλπ). Η μόνη εντολή που δε χρησιμοποιεί την ALU είναι η b (branch).

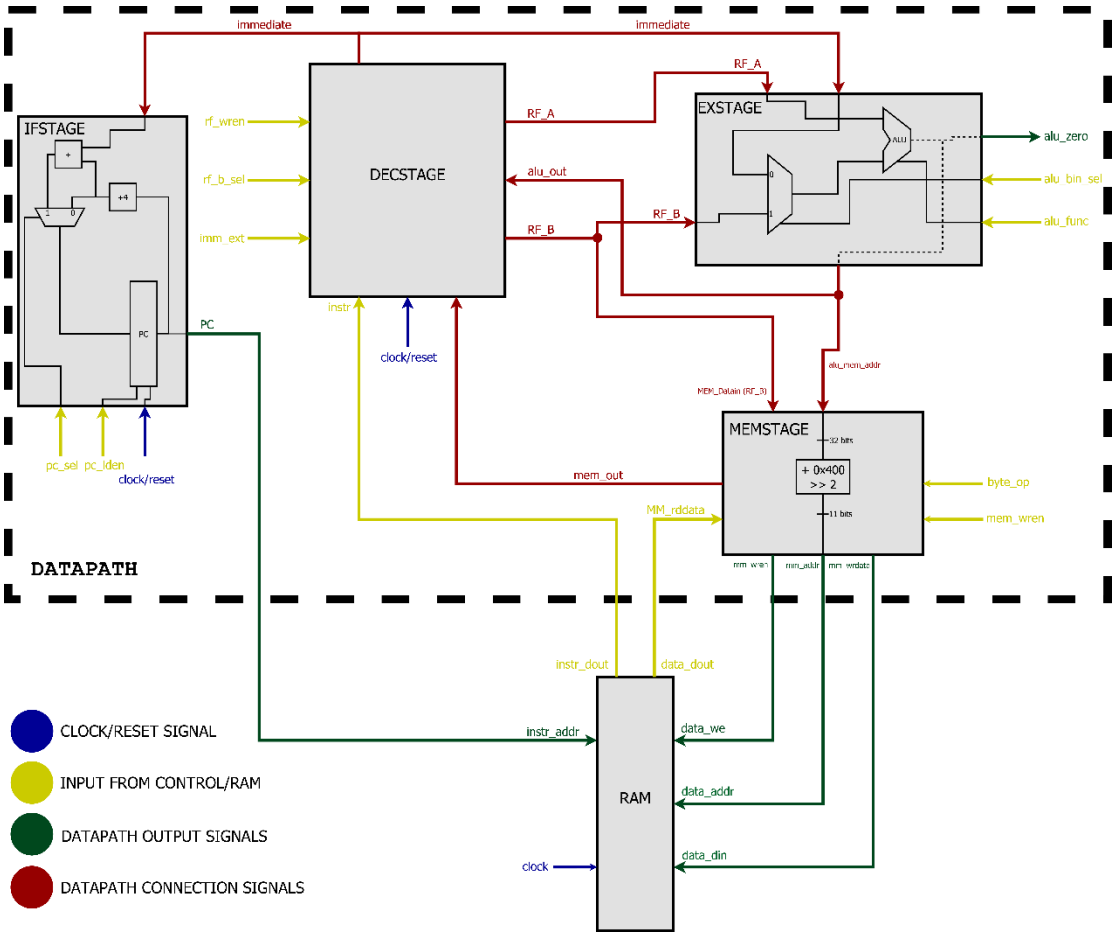
Το στάδιο MEMSTAGE είναι υπεύθυνο για την προσπέλαση της μνήμης RAM. Η RAM χωρίζεται σε 2 τμήματα στο code και στο data segment.

ΤΡΙΤΗ ΦΑΣΗ:

Τα 4 μεγάλα components που περιγράφονται παραπάνω ενώθηκαν και δημιουργήθηκε το DATAPATH. Παράλληλα με το DATAPATH δημιουργήθηκε και η μνήμη RAM μεγέθους 2048x4 bytes καθώς και το CONTROL. Το CONTROL τροφοδοτεί σχεδόν όλες τις εισόδους του DATAPATH. Παρακάτω τα Block Diagrams του DATAPATH και του τελικού επεξεργαστή.



Block Diagram processor single cycle



Block Diagram DATAPATH with RAM