



**Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης**  
**Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών**  
**Υπολογιστών**  
**Ψηφιακά συστήματα HW-1**

**ΧΕΙΜΕΡΙΝΟ ΕΞΑΜΗΝΟ 2021-2022**

**Project - Μέρος 2**

**Σχεδίαση Βασικών Βαθμίδων του Datapath ενός**  
**Απλού Επεξεργαστή**

**Σκοπός αυτού του μέρους:**

1. Ο ορισμός της αρχιτεκτονικής συνόλου εντολών που θα χρησιμοποιήσετε και στο επόμενο μέρος.
2. Η σχεδίαση της βαθμίδας ανάκλησης εντολών (IF).
3. Η σχεδίαση της βαθμίδας αποκωδικοποίησης εντολών (DECODE).
4. Η σχεδίαση της βαθμίδας Εκτέλεσης Εντολών (ALU).
5. Η σχεδίαση της βαθμίδας Πρόσβασης Μνήμης (MEM).

**Αρχιτεκτονική Συνόλου Εντολών**

Καλείστε να υλοποιήσετε διάφορα τμήματα ενός non-pipelined επεξεργαστή βασισμένου σε υποσύνολο της αρχιτεκτονικής συνόλου εντολών το οποίο αποτελείται από τα εξής στοιχεία:

1. 32 καταχωρητές των 32 bits. Ο καταχωρητής R0 είναι πάντα μηδέν.
2. 32 bit πλάτος εντολών με μέγεθος και θέση πεδίων που περιγράφονται παρακάτω.
3. Εντολές αριθμητικών και λογικών πράξεων: add, sub, and, not, or, shr, shl, sla, rol, ror, li, addi, andi, ori.
4. Εντολές διακλάδωσης: b, beq, bneq.
5. Εντολές μνήμης: lb, sb, lw, sw.

Οι παραπάνω εντολές έχουν δύο τύπους format:

6-bits	5-bits	5-bits	5-bits	5-bits	6-bits
Opcode	rs	rd	rt	not-used	func

6-bits	5-bits	5-bits	16-bits
Opcode	rs	rd	Immediate

Η διευθυνσιοδότηση της μνήμης γίνεται με διευθύνσεις byte, και οι εντολές και τα δεδομένα (των εντολών lb, sb, lw και sw) πρέπει να είναι ευθυγραμμισμένα σε πολλαπλάσια των 4 bytes.

Η κωδικοποίηση των εντολών γίνεται σύμφωνα με τον ακόλουθο πίνακα:

Opcode	FUNC	ΕΝΤΟΛΗ	ΠΡΑΞΗ
100000	110000	add	$RF[rd] \leftarrow RF[rs] + RF[rt]$
100000	110001	sub	$RF[rd] \leftarrow RF[rs] - RF[rt]$
100000	110010	and	$RF[rd] \leftarrow RF[rs] \& RF[rt]$
100000	110100	not	$RF[rd] \leftarrow ! RF[rs]$
100000	110011	or	$RF[rd] \leftarrow RF[rs]   RF[rt]$
100000	111000	sra	$RF[rd] \leftarrow RF[rs] \gg 1$
100000	111001	sll	$RF[rd] \leftarrow RF[rs] \ll 1$ (Logical, zero fill LSB)
100000	111010	srl	$RF[rd] \leftarrow RF[rs] \gg 1$ (Logical, zero fill MSB)
100000	111100	rol	$RF[rd] \leftarrow \text{Rotate left}(RF[rs])$
100000	111101	ror	$RF[rd] \leftarrow \text{Rotate right}(RF[rs])$
111000	-	li	$RF[rd] \leftarrow \text{SignExtend}(\text{Imm})$
111001	-	lui	$RF[rd] \leftarrow \text{Imm} \ll 16$ (zero-fill)
110000	-	addi	$RF[rd] \leftarrow RF[rs] + \text{SignExtend}(\text{Imm})$
110010	-	andi	$RF[rd] \leftarrow RF[rs] \& \text{ZeroFill}(\text{Imm})$
110011	-	ori	$RF[rd] \leftarrow RF[rs]   \text{ZeroFill}(\text{Imm})$
111111	-	b	$PC \leftarrow PC + 4 + (\text{SignExtend}(\text{Imm}) \ll 2)$
000000	-	beq	if ( $RF[rs] == RF[rd]$ ) $PC \leftarrow PC + 4 + (\text{SignExtend}(\text{Imm}) \ll 2)$ else $PC \leftarrow PC + 4$
000001	-	bne	if ( $RF[rs] != RF[rd]$ ) $PC \leftarrow PC + 4 + (\text{SignExtend}(\text{Imm}) \ll 2)$ else $PC \leftarrow PC + 4$

Opcode	FUNC	ΕΝΤΟΛΗ	ΠΡΑΞΗ
000011	-	lb	$RF[rd] \leftarrow ZeroFill(31 \text{ downto } 8) \& MEM[RF[rs] + SignExtend(Imm)](7 \text{ downto } 0)$
000111	-	sb	$MEM[RF[rs] + SignExtend(Imm)] \leftarrow ZeroFill(31 \text{ downto } 8) \& RF[rd] (7 \text{ downto } 0)$
001111	-	lw	$RF[rd] \leftarrow MEM[RF[rs] + SignExtend(Imm)]$
011111	-	sw	$MEM[RF[rs] + SignExtend(Imm)] \leftarrow RF[rd]$

Το ZeroFill συμπληρώνει τα bits με 0, ενώ το SignExtend αντιγράφει το 15ο bit του Imm στα 16-31 bits.

## Διεξαγωγή

### A. Μελετήστε την κωδικοποίηση των εντολών του επεξεργαστή

Μελετήστε την κωδικοποίηση των εντολών. Παρατηρήστε την ομαδοποίηση τους, έχοντας στο μυαλό σας τους γενικούς στόχους της αποκωδικοποίησης: η παραγωγή όλων των απαραίτητων σημάτων ελέγχου για την εκτέλεση της εντολής. Τέτοια σήματα είναι ο κωδικός πράξης της ALU, οι διευθύνσεις ανάγνωσης και τυχόν εγγραφής στην Register File, η επιλογή δύο καταχωρητών ή καταχωρητή και Immediate για μια πράξη, κ.α.

Η ανάθεση κωδικών έγινε έτσι ώστε η αποκωδικοποίηση των εντολών να είναι μια σχετικά απλή διαδικασία. Βρείτε τις υπάρχουσες συμμετρίες ώστε να απλοποιήσετε την λογική αποκωδικοποίησης.

### B. Σχεδιασμός και υλοποίηση βαθμίδας ανάκλησης εντολών (IF)

#### Παραγωγή Μνήμης 1024x32

Χρησιμοποιήστε τον κώδικα που περιγράφεται στην Εικόνα 1 για να φτιάξετε μία μνήμη ROM 1024 θέσεων των 32 bits. Η μνήμη πρέπει να έχει μία θύρα ανάγνωσης. Η διεπαφή της μνήμης πρέπει να έχει τα εξής σήματα: διεύθυνση ανάγνωσης, δεδομένα που διαβάστηκαν και ρολόι. Η διεπαφή της μνήμης συνοψίζεται στον ακόλουθο πίνακα:

Σήμα	Πλάτος	Λειτουργία
Addr	10 bits	Διεύθυνση για ανάγνωση εντολής
Dout	32 bits	Δεδομένα που διαβάστηκαν από την μνήμη
Clk	1bit	Ρολόι

```

module IFSTAGE(
    input clk,
    input [9:0] addr,
    output reg [31:0] dout
);

reg [31:0] ROM [1023:0];

initial
begin
    $readmemb("rom.data", ROM);
end

always @(posedge clk)
begin
    dout <= ROM[addr];
end

endmodule

```

Εικόνα 1: Δημιουργία μνήμης ROM

**Προσοχή!!** Για να αρχικοποιήσετε την μνήμη χρειάζεται ο κώδικας που είναι με έντονα γράμματα στην παραπάνω εικόνα. Ο συγκεκριμένος κώδικας διαβάζει το αρχείο `rom.data` (το οποίο θα πρέπει να βρίσκεται στον ίδιο φάκελο με το `project` σας) και φορτώνει κάθε γραμμή του αρχείου σε μία νέα διεύθυνση.

### **B1. Σχεδίαση των επιμέρους τμημάτων και υλοποίηση της βαθμίδας**

Χρησιμοποιώντας ένα αντίγραφο της μνήμης αυτής και άλλη λογική, σχεδιάστε και υλοποιήστε μια βαθμίδα ανάκλησης εντολών. Η βαθμίδα είναι απλή και αποτελείται από τα παρακάτω δομικά στοιχεία:

1. Τον καταχωρητή PC (32 bits).
2. Την μνήμη εντολών που σχεδιάσατε παραπάνω
3. Έναν αθροιστή (ή ακριβέστερα αυξητή/incrementor κατά 4 ο οποίος υπολογίζει την τιμή  $PC + 4$ ).
4. Έναν αθροιστή ο οποίος υπολογίζει την τιμή  $(PC + 4) + Immediate$  για εντολές διακλάδωσης.
5. Ένα πολυπλέκτη 2 σε 1 ο οποίος διαλέγει μία από τις 2 δυνατές τιμές ( $PC+4$ ,  $PC+4+Immediate$ ) για να ενημερωθεί ο PC.

Η διεπαφή της βαθμίδας αυτής συνοψίζεται ως εξής:

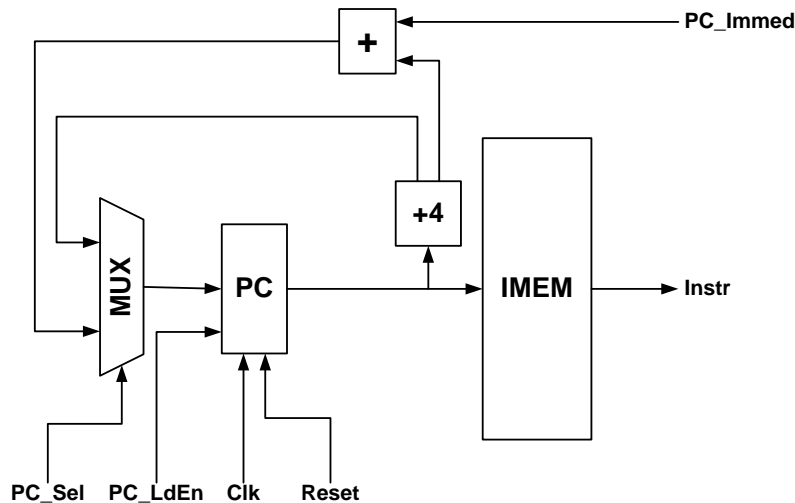
Σήμα	Πλάτος	Είδος	Περιγραφή
PC_Immed	32 bits	Είσοδος	Τιμή Immediate για εντολές b, beq, bne
PC_sel	1 bit	Είσοδος	Επιλογή εισόδου για ενημέρωση του PC:  0 → $PC+4$ ,  1 → $PC+4 + Immediate$ .
PC_LdEn	1 bit	Είσοδος	Ενεργοποίηση εγγραφής στον PC
Reset	1 bit	Είσοδος	Είσοδος Reset για αρχικοποίηση του καταχωρητή PC
Clk	1 bit	Είσοδος	Ρολόι
Instr	32 bits	Εξοδος	Η εντολή που ανακλήθηκε.

### Εκτέλεση

1. Παρατηρήστε το διάγραμμα της βαθμίδας στο σχήμα που ακολουθεί ώστε να κατανοήσετε την λειτουργία της.
2. Γράψτε τον κώδικα Verilog που να υλοποιεί τα επιμέρους τμήματα της βαθμίδας και κάντε τις κατάλληλες εσωτερικές συνδέσεις για να υλοποιήσετε την βαθμίδα ανάκλησης εντολών χρησιμοποιώντας τη μνήμη που παράγατε παραπάνω, πολυπλέκτες, καταχωρητές και ό,τι άλλη λογική χρειάζεστε. Ονομάστε το αρχείο σας IFSTAGE.v
3. Προσομοιώστε και επιβεβαιώστε την βαθμίδα ανάκλησης εντολών. Χρησιμοποιήστε το αρχείο rom.data για να αρχικοποιήσετε τη μνήμη και ελέγξτε την λειτουργία της βαθμίδας ανάκλησης σε ακολουθιακές προσβάσεις αλλά και ενεργοποιώντας τις άλλες τιμές για εγγραφή στον PC.

### Παρατήρηση

1. Προσέξτε ότι ο PC αυξάνεται κατά 4 για να δώσει την επόμενη εντολή, ενώ ο κώδικας που σας δίνεται, το addr του IMEM, δίνει 1 εντολή ανά 1 αλλαγή στη διεύθυνση μνήμης. Για το λόγο αυτό χρησιμοποιήστε τα [11:2] bits του PC στην είσοδο του IMEM.



**Σχήμα 1:** Σχηματικό διάγραμμα βαθμίδας ανάκλησης εντολών.

### Γ. Σχεδιασμός και υλοποίηση βαθμίδας αποκωδικοποίησης εντολών (DECODE)

Χρησιμοποιώντας ένα αντίγραφο του αρχείου καταχωρητών που παράγατε στο **1<sup>ο</sup> μέρος** και άλλη λογική, σχεδιάστε και υλοποιήστε μια βαθμίδα αποκωδικοποίησης εντολών. Η βαθμίδα είναι αρκετά απλή στην λογική και αποτελείται από:

1. Το αρχείο καταχωρητών
2. Έναν πολυπλέκτη 2 σε 1 ο οποίος επιλέγει μία από τις 2 προελεύσεις των δεδομένων προς εγγραφή στο αρχείο καταχωρητών (ALU, MEM)
3. Μία μονάδα που δέχεται σαν είσοδο τα 16 bits που αποτελούν το immediate μιας εντολής και το μετατρέπει σε ένα σήμα 32 bits επιλέγοντας αν θα γίνει αριστερή ολίσθηση του immediate κατά 2 ή όχι, **και** επίσης αν θα γίνει zero-filling ή sign-extension του immediate προκειμένου να μετατραπεί σε 32 bit.

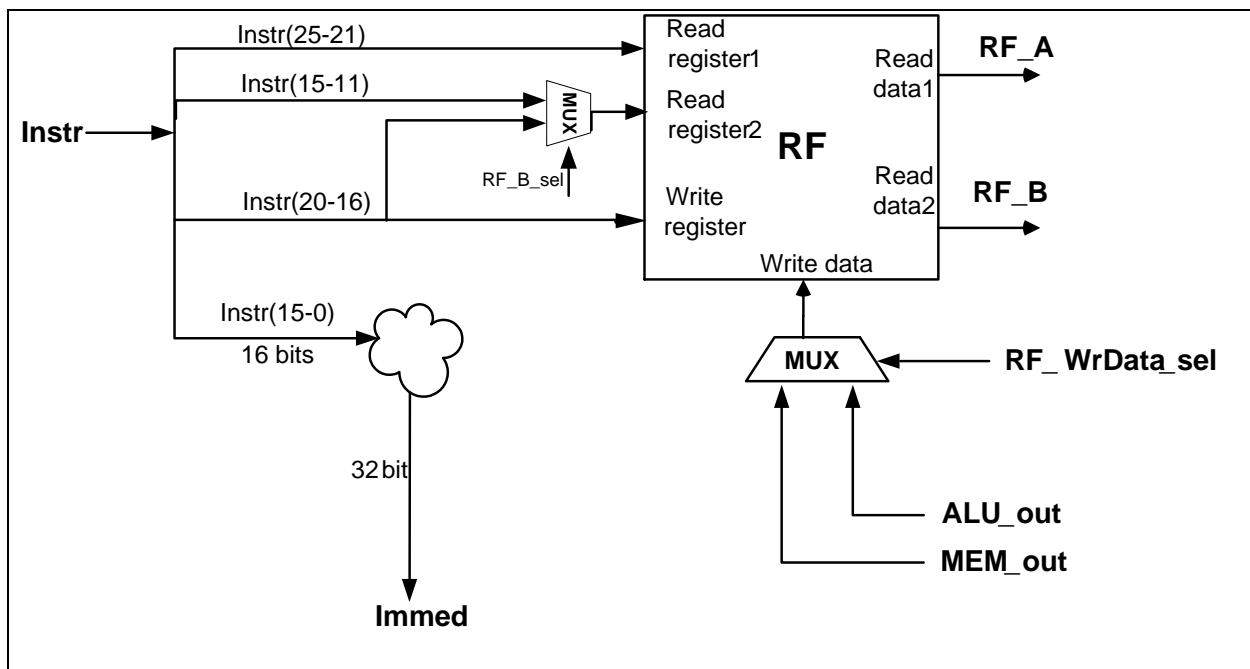
Η διεπαφή της βαθμίδας συνοψίζεται ως εξής:

Σήμα	Πλάτος	Είδος	Περιγραφή
Instr	32 bit	Είσοδος	Η εντολή που πρέπει να αποκωδικοποιηθεί
RF_WrEn	1 bit	Είσοδος	Ενεργοποίηση εγγραφής καταχωρητή
ALU_out	32 bits	Είσοδος	Δεδομένα εγγραφής καταχωρητή προερχόμενα από την ALU
MEM_out	32 bits	Είσοδος	Δεδομένα εγγραφής καταχωρητή προερχόμενα από τη μνήμη
RF_WrData_sel	1 bit	Είσοδος	Επιλογή πεδίου που καθορίζει την προέλευση δεδομένων προς εγγραφή: 1 → MEM 0 → ALU

Σήμα	Πλάτος	Είδος	Περιγραφή
RF_B_sel	1 bit	Είσοδος	Επιλογή πεδίου που καθορίζει τον δεύτερο καταχωρητή ανάγνωσης:  0 → Instr(15-11)  1 → Instr(20-16)
Clk	1 bit	Είσοδος	Ρολόι
Immed	32 bits	Έξοδος	Immediate προς τις επόμενες βαθμίδες
RF_A	32 bits	Έξοδος	Η τιμή του 1 <sup>ου</sup> καταχωρητή
RF_B	32 bits	Έξοδος	Η τιμή του 2 <sup>ου</sup> καταχωρητή

### Εκτέλεση

1. Παρατηρήστε το διάγραμμα της βαθμίδας στο σχήμα που ακολουθεί ώστε να κατανοήσετε την λειτουργία της
2. Ο καταχωρητής R0 πρέπει να έχει πάντα την τιμή "0" (μηδέν). Πώς το υλοποιείτε αυτό;
3. Γράψτε τον κώδικα Verilog που να υλοποιεί τα επιμέρους τμήματα της βαθμίδας και κάντε τις κατάλληλες εσωτερικές συνδέσεις για να υλοποιήσετε την βαθμίδα ανάκλησης εντολών χρησιμοποιώντας τη αρχείο Register File που παράγατε στο πρώτο εργαστήριο, πολυπλέκτες, καταχωρητές και ό,τι άλλη λογική χρειάζεστε. Ονομάστε το αρχείο σας: DECSTAGE.v
4. Προσομοιώστε την βαθμίδα αποκωδικοποίησης εντολών καλύπτοντας τις βασικές κατηγορίες εντολών ώστε να επιβεβαιώσετε την λογική αποκωδικοποίησης. Επεκτείνετε την προσομοίωση για να καλύψετε όσο το δυνατόν περισσότερες περιπτώσεις.



**Σχήμα 2:** Σχηματικό διάγραμμα βαθμίδας αποκωδικοποίησης εντολών.

#### Δ. Σχεδιασμός και υλοποίηση βαθμίδας Εκτέλεσης Εντολών (ALU)

Χρησιμοποιώντας την ALU που σχεδιάσατε στο 1<sup>ο</sup> μέρος και άλλη λογική, σχεδιάστε και υλοποιήστε μια βαθμίδα εκτέλεσης αριθμητικών και λογικών εντολών. Η βαθμίδα αποτελείται από:

1. Την ALU
2. Ένας πολυπλέκτης που επιλέγει ποιος θα είναι ο δεύτερος τελεστής της ALU.

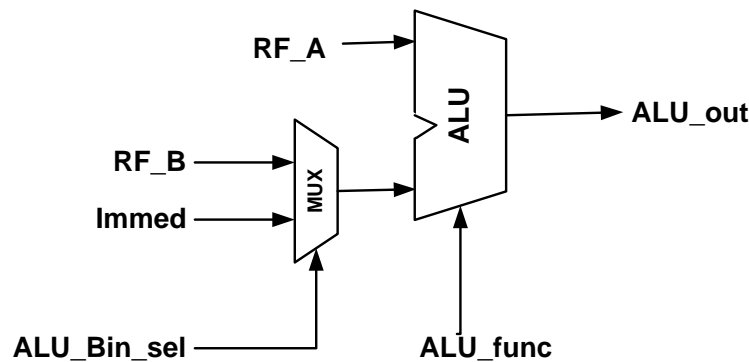
Συνοπτικά η διεπαφή της βαθμίδας Εκτέλεσης Εντολών (ALU) είναι η εξής:

Σήμα	Πλάτος	Είδος	Περιγραφή
RF_A	32 bits	Είσοδος	RF[rs]
RF_B	32 bits	Είσοδος	RF[rt] ή RF[rd]
Immed	32 bits	Είσοδος	Immediate
ALU_Bin_sel	1 bit	Είσοδος	Επιλογή Εισόδου B της ALU από RF_B ή Immediate 1 → Immed 0 → RF_B
ALU_func	4 bit	Είσοδος	Πράξη ALU
ALU_out	32 bit	Έξοδος	Αποτέλεσμα ALU



## Εκτέλεση

1. Παρατηρήστε το διάγραμμα της βαθμίδας στο σχήμα που ακολουθεί ώστε να κατανοήσετε την λειτουργία της
2. Γράψτε τον κώδικα Verilog που να υλοποιεί τα επιμέρους τμήματα της βαθμίδας και κάντε τις κατάλληλες εσωτερικές συνδέσεις για να υλοποιήσετε την βαθμίδα ανάκλησης εντολών χρησιμοποιώντας τη μνήμη που παράγατε παραπάνω, πολυπλέκτες, καταχωρητές και ό,τι άλλη λογική χρειάζεστε. Ονομάστε το αρχείο σας:ALUSTAGE.v
3. Προσομοιώστε και επιβεβαιώσετε την βαθμίδα εκτέλεσης αριθμητικών και λογικών εντολών.



**Σχήμα 3:** Σχηματικό διάγραμμα βαθμίδας εκτέλεσης εντολών.

### Ε. Σχεδιασμός και υλοποίηση βαθμίδας Πρόσβασης Μνήμης (MEM)

Χρησιμοποιήστε τον κώδικα που περιγράφεται στην Εικόνα 2 για να παράγετε μια μνήμη RAM 1024 θέσεων των 32 bits. Η μνήμη έχει μία θύρα ανάγνωσης και εγγραφής.

Η διεπαφή της βαθμίδας πρέπει να έχει τις εξής εισόδους και εξόδους:

Σήμα	Πλάτος	Είδος	Περιγραφή
clk	1 bit	Είσοδος	Ρολόι
Mem_WrEn	1 bit	Είσοδος	Σημαία ενεργοποίησης εγγραφής στη μνήμη
ALU_MEM_Addr	32 bits	Είσοδος	Αποτέλεσμα ALU (βλέπε εντολές lb, sb, lw, sw)
MEM_DataIn	32 bits	Είσοδος	Αποτέλεσμα RF[rd] για αποθήκευση στη μνήμη για εντολές swap και sb, sw
MEM_DataOut	32 bits	Έξοδος	Δεδομένα που φορτώθηκαν από τη μνήμη προς εγγραφή σε καταχωρητή για εντολές lb, lw

```

module MEMSTAGE(
    input clk,
    input we,
    input [9:0] addr,
    input [31:0] din,
    output reg [31:0] dout
);
reg [31:0] RAM [1023:0];
always @(posedge clk)
begin
    if(we)
        RAM[addr] = din;
    else
        dout = RAM[addr];
end
endmodule

```

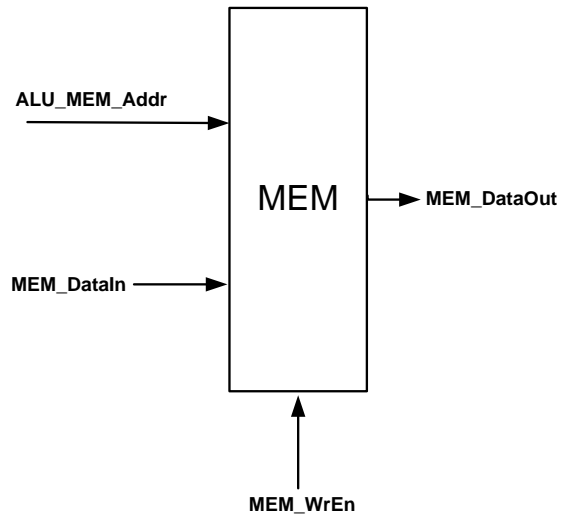
Εικόνα 2: Δημιουργία μνήμης RAM

### Εκτέλεση

1. Παρατηρήστε το διάγραμμα της βαθμίδας στο σχήμα που ακολουθεί ώστε να κατανοήσετε την λειτουργία της.
2. Γράψτε τον κώδικα Verilog που υλοποιεί την βαθμίδα Πρόσβασης Μνήμης χρησιμοποιώντας τη μνήμη που παράγατε παραπάνω. Ονομάστε το αρχείο σας MEMSTAGE.v
3. Προσομοιώστε και επιβεβαιώσετε την βαθμίδα πρόσβασης στη μνήμη.

### Παρατήρηση

1. Προσέξτε ότι ο κώδικας που σας δίνεται για το addr του MEM, είναι 10 bits, ενώ η έξοδος της ALU είναι 32 bits. Για το λόγο αυτό χρησιμοποιήστε τα [11:2] bits της ALU στο addr του MEM.



Σχήμα 4: Σχηματικό διάγραμμα βαθμίδας πρόσβασης στη μνήμη.

#### Παρατηρήσεις / Σημειώσεις

- Στον ορισμό των modules βάλτε τις παραμέτρους με την σειρά που εμφανίζονται στους πίνακες.