CPU Architecture

LAB3 assignment

Digital System Design with VHDL

(Multi-cycle CPU design)

Hanan Ribo

09/06/2024

Table of contents

1. Aim of the Task:

- System design using concurrent and sequential logic principles using advanced Simulation methods (based on material given in LAB1 and LAB2 tasks).
- Controller design based on methodology of Control and Datapath separation.
- Preparation for LAB4 task – FPGA based design synthesis of a given design.
- Proper analysis and understanding of architecture design.
- Performing and understanding functional validation and verification of an architecture design.

2. Assignment definition:

In this task, you are required to design a controller-based processing machine as a Multi-cycle CPU in order to run a given program code. The preparation material for this lab has been learned in the prerequisites course “Central Processing Unit Architecture - theory” until lecture five and includes in addition to the self-leaning material given on a subjects of FSM methodology design and advanced functional verification.

3. Controller based system:

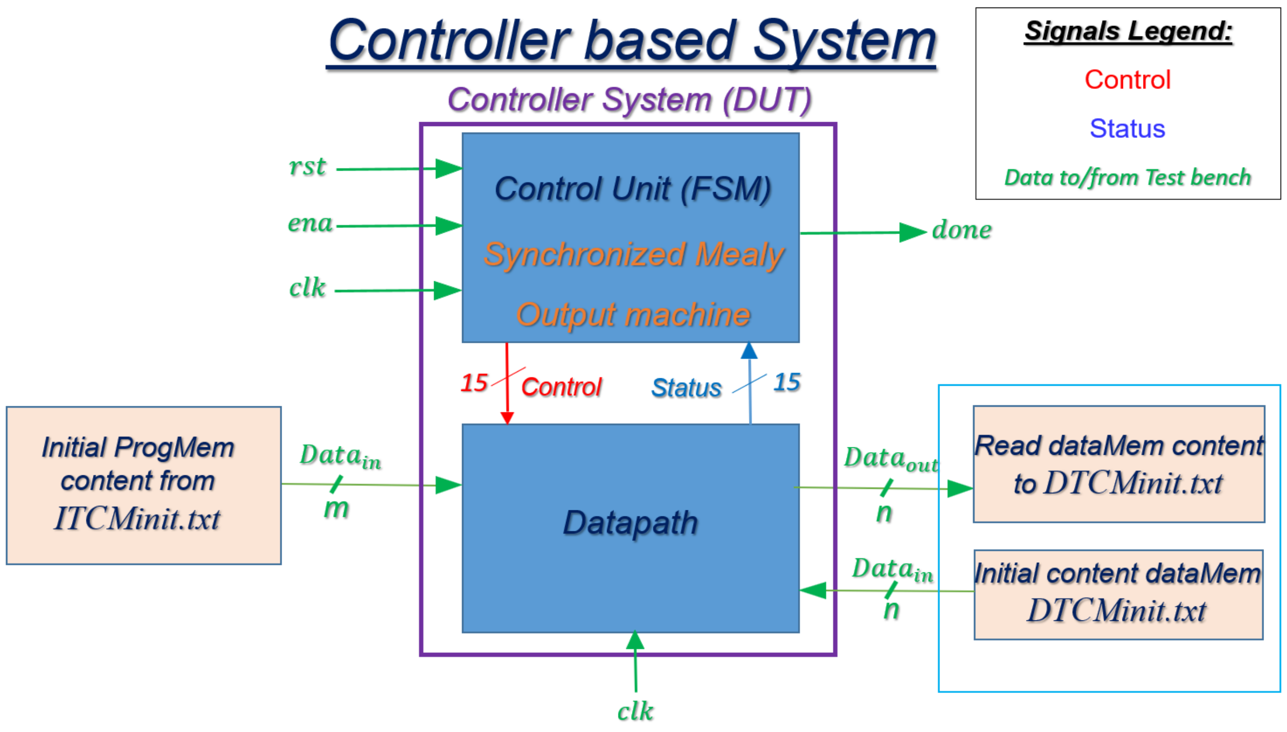


Figure 1: Overall DUT structure

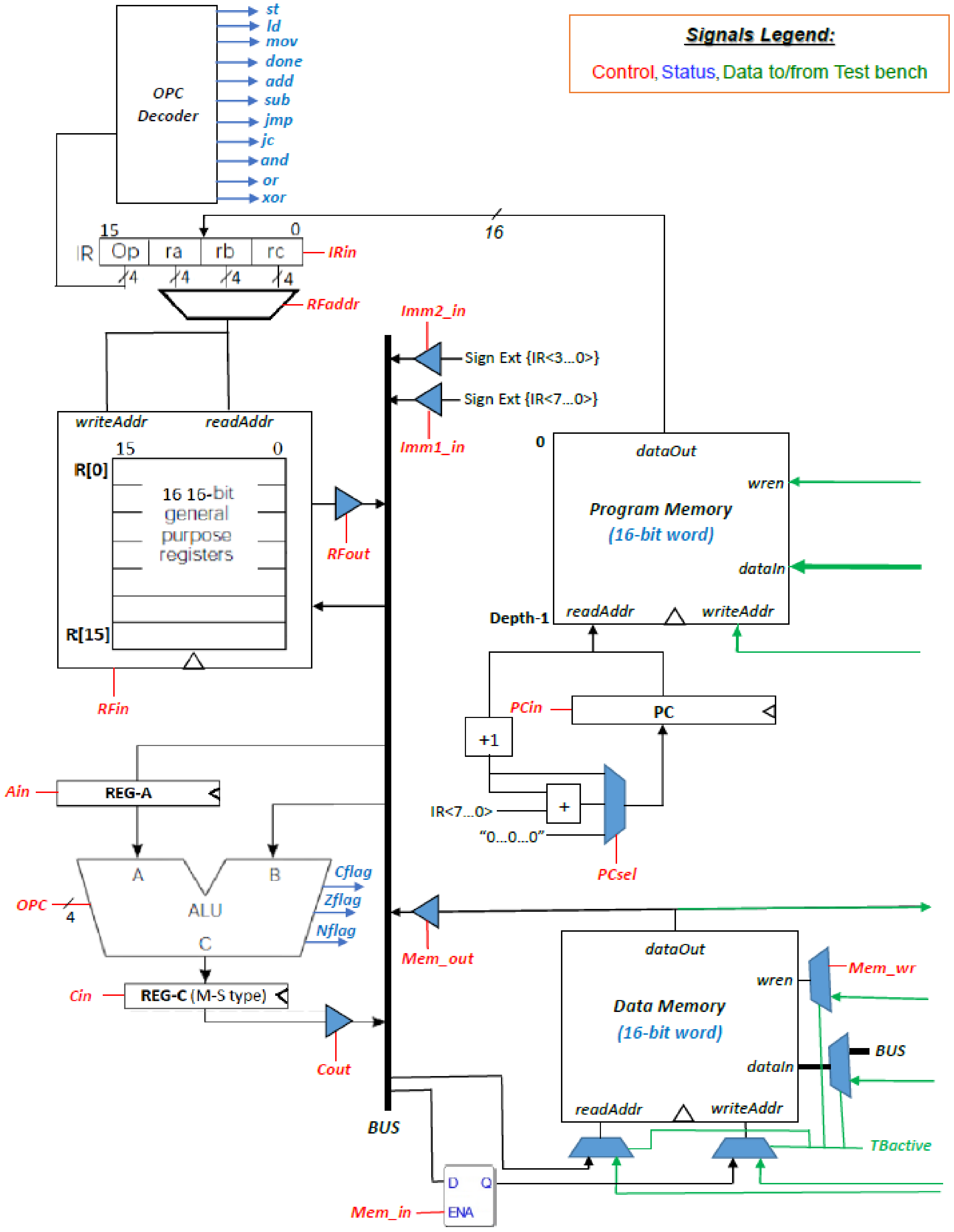


Figure 2: Datapath structure

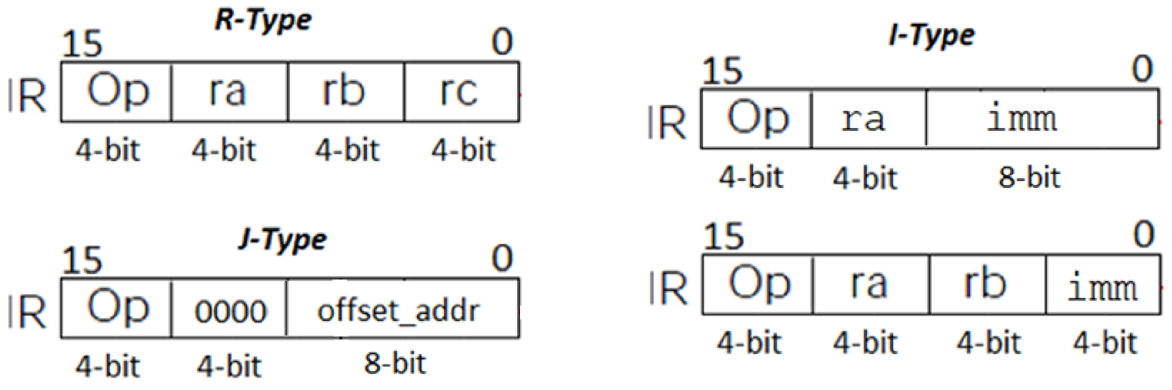


Figure 3: Instruction Formats

Instruction Format	Decimal value	OPC	Instruction	Explanation	N	Z	C
R-Type	0	0000	add ra,rb,rc	R[ra]<=R[rb]+R[rc]	*	*	*
			nop	R[0]<=R[0]+R[0] (*emulated instruction*)	*	*	*
	1	0001	sub ra,rb,rc	R[ra]<=R[rb]-R[rc]	*	*	*
	2	0010	and ra,rb,rc	R[ra]<=R[rb] and R[rc]	*	*	-
	3	0011	or ra,rb,rc	R[ra]<=R[rb] or R[rc]	*	*	-
	4	0100	xor ra,rb,rc	R[ra]<=R[rb] xor R[rc]	*	*	-
	5	0101	***unused***				
	6	0110	***unused***				
J-Type	7	0111	jmp offset_addr	PC<=PC+1+offset_addr	-	-	-
	8	1000	jc /jhs offset_addr	If(Cflag==1) PC<=PC+1+offset_addr	-	-	-
	9	1001	jnc /jlo offset_addr	If(Cflag==0) PC<=PC+1+offset_addr	-	-	-
	10	1010	***unused***				
	11	1011	***unused***				
I-Type	12	1100	mov ra,imm	R[ra]<=imm	-	-	-
	13	1101	ld ra,imm(rb)	R[ra] <= M[imm+R[rb]]	-	-	-
	14	1110	st ra,imm(rb)	M[imm+R[rb]] <= R[ra]	-	-	-
	15	1111	done	Signals the TB to read the DTCM content	-	-	-

Note: * The status flag bit is affected , - The status flag bit is not affected

Table 1 : Multi-cycle CPU ISA

Control Unit

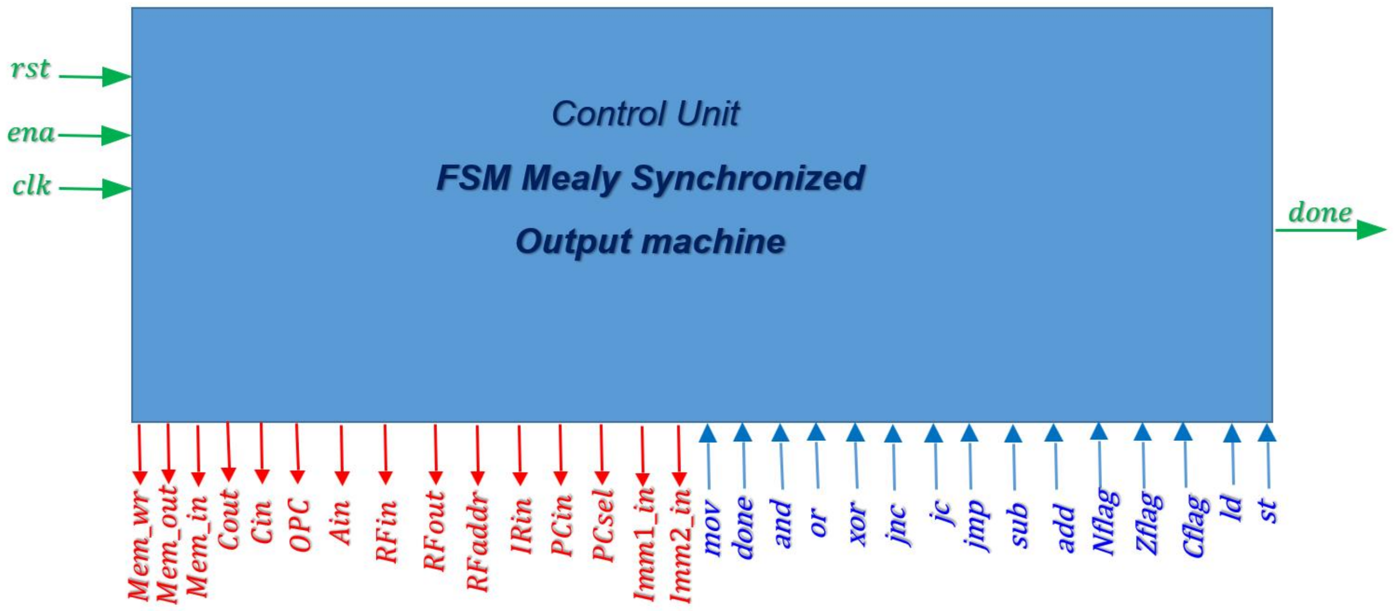


Figure 4: Control unit structure

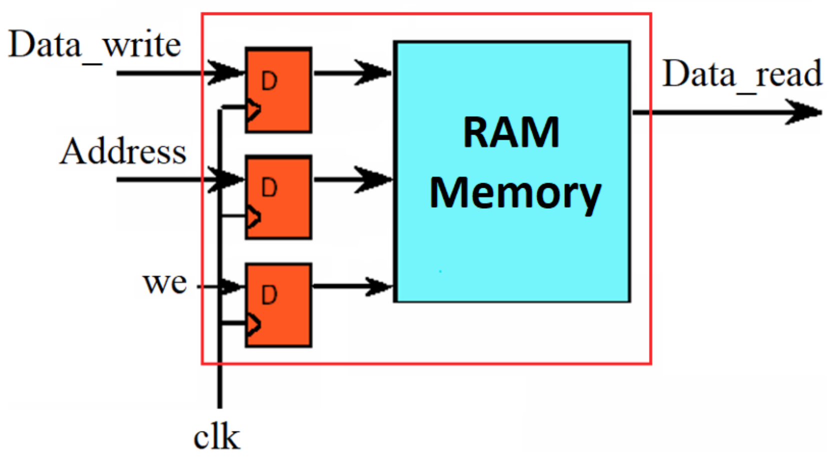


Figure 5: Single Port RAM with an unregistered output (this structure used for Register File and Program Memory with a given VHDL code)

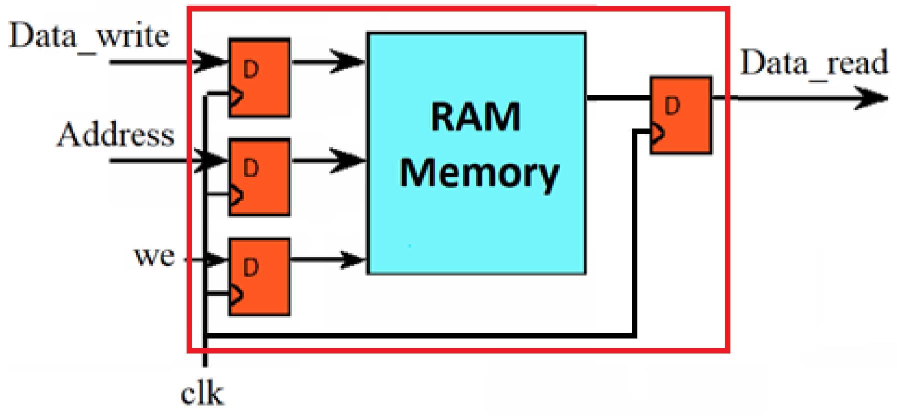
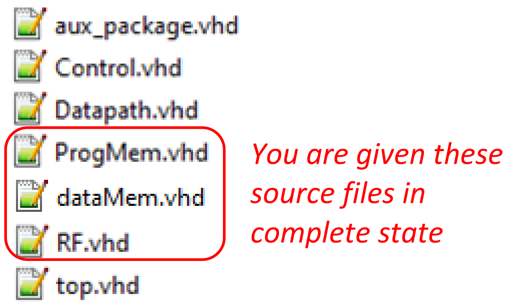


Figure 6: Single Port RAM with a registered output (this structure used for Data Memory with a given VHDL code)

- The given files <u>that you must use in your project</u>:

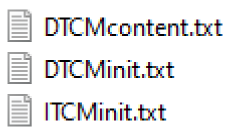


aux_package.vhd
Control.vhd
Datapath.vhd
ProgMem.vhd
dataMem.vhd
RF.vhd
top.vhd

- ✓ You must use the listed above source files, you are required to fill them up yourself**.**
- ✓ You can add additional VHDL source code files.
- ✓ Other entities can be designed and modeled behaviorally, structurally, etc.

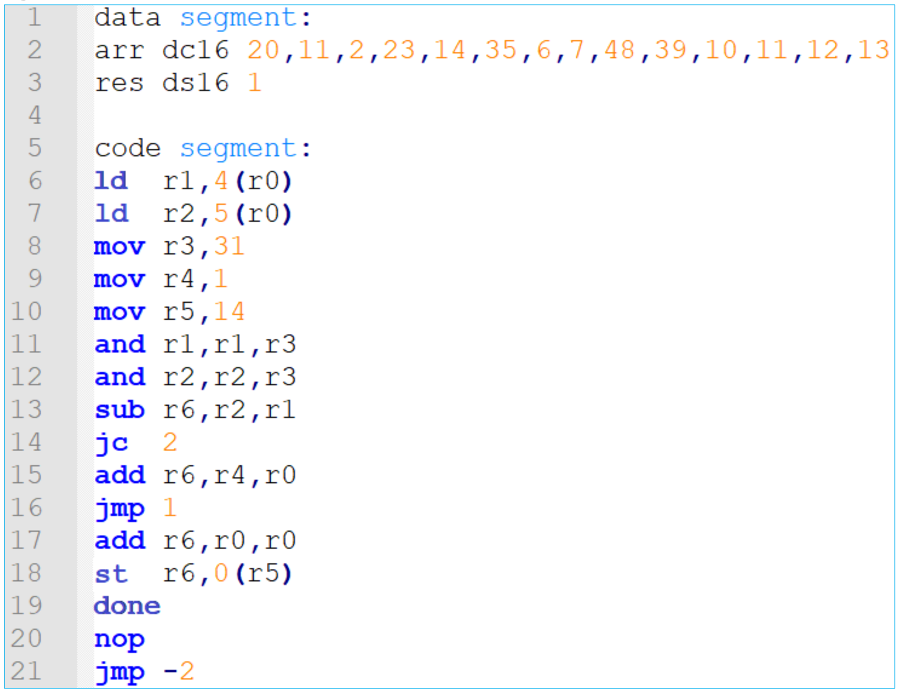
4. File based simulation of DUT:

- As depicted in figure 1, in order to simulate the system DUT we use file based simulation.
- <u>Two Input and one Output files structure (according the above given figure at clause 3)</u>:



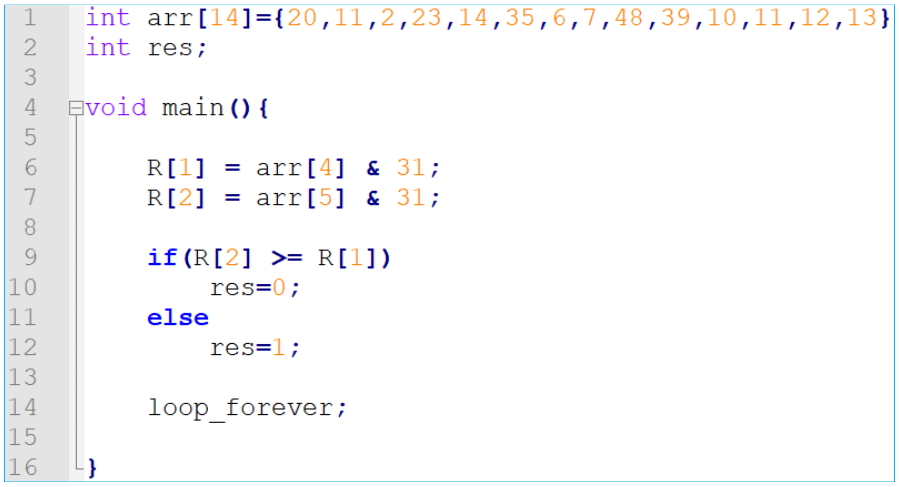
DTCMcontent.txt
DTCMinit.txt
ITCMinit.txt

5. Running Code Example:



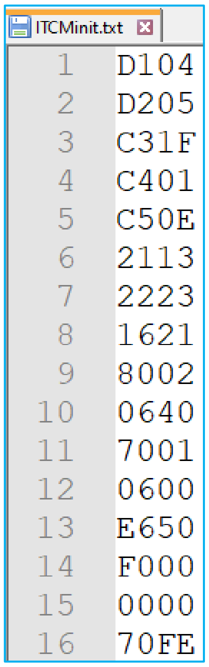
```
data segment:
arr dc16 20,11,2,23,14,35,6,7,48,39,10,11,12,13
res ds16 1

code segment:
ld  r1,4(r0)
ld  r2,5(r0)
mov r3,31
mov r4,1
mov r5,14
and r1,r1,r3
and r2,r2,r3
sub r6,r2,r1
jc  2
add r6,r4,r0
jmp 1
add r6,r0,r0
st  r6,0(r5)
done
nop
jmp -2
```

Assembly code



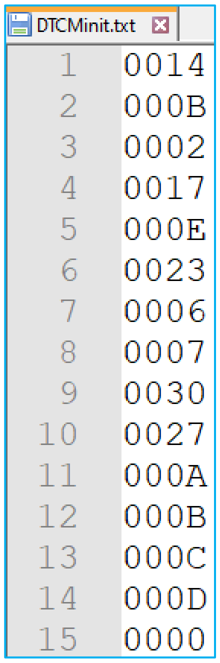
```
int arr[14]={20,11,2,23,14,35,6,7,48,39,10,11,12,13}
int res;

void main(){

    R[1] = arr[4] & 31;
    R[2] = arr[5] & 31;

    if(R[2] >= R[1])
        res=0;
    else
        res=1;

    loop_forever;

}
```

Equivalent Pseudo code



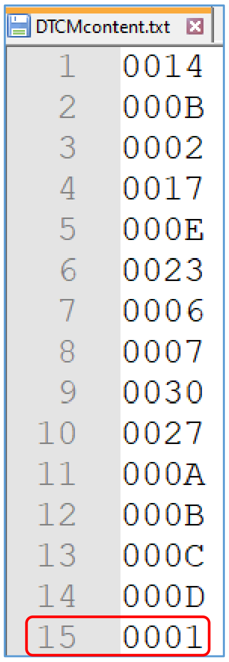
ITCMinit.txt

```
D104
D205
C31F
C401
C50E
2113
2223
1621
8002
0640
7001
0600
E650
F000
0000
70FE
```



DTCMinit.txt

```
0014
000B
0002
0017
000E
0023
0006
0007
0030
0027
000A
000B
000C
000D
0000
```



DTCMcontent.txt

```
0014
000B
0002
0017
000E
0023
0006
0007
0030
0027
000A
000B
000C
000D
0001
```

Updated values after program run

6. Requirements:

1. The design must be well commented.
2. Elaborated analysis and wave forms:
   - Remove irrelevant signals.
   - Zoom on regions of interest.
   - Draw clouds on the waveform with explanations of what is happening (Figure 4).
   - Change the waveform colors in ModelSim for clear documentation **(Tools->Edit Preferences->Wave Windows).**

3. A ZIP file in the form of **id1_id2.zip** (where id1 and id2 are the identification number of the submitters, and id1 < id2) *must be upload to Moodle only by student with id1* (any of these rules violation disqualifies the task submission).

4. The **ZIP** file will contain (***only the exact next sub folders***):

Directory	**Contains**	**Comments**
DUT	Project VHDL files	**Only VHDL files of DUT**, excluding test bench **Note: your project files must be well compiled without errors as a basic condition before submission**
TB	VHDL files that are used for test bench	One **tb.vhd** for the Datapath One **tb.vhd** for the Control unit One **tb.vhd** for the overall DUT
SIM	ModelSim DO files (wave, list)	One **tb.vhd** for the Datapath One **tb.vhd** for the Control unit One **tb.vhd** for the overall DUT (with Datatpath and Control unit signals included)
DOC	Project documentation	• ***readme.txt*** (list of the DUT **.vhd* files with their brief functional description) • ***pre3.pdf*** (report file that includes brief explanation of the top module with its wave diagrams as shown in figure 4) • ***designGraph.pdf*** (elaborated FSM graph of your DUT)

Table 2: Directory Structure

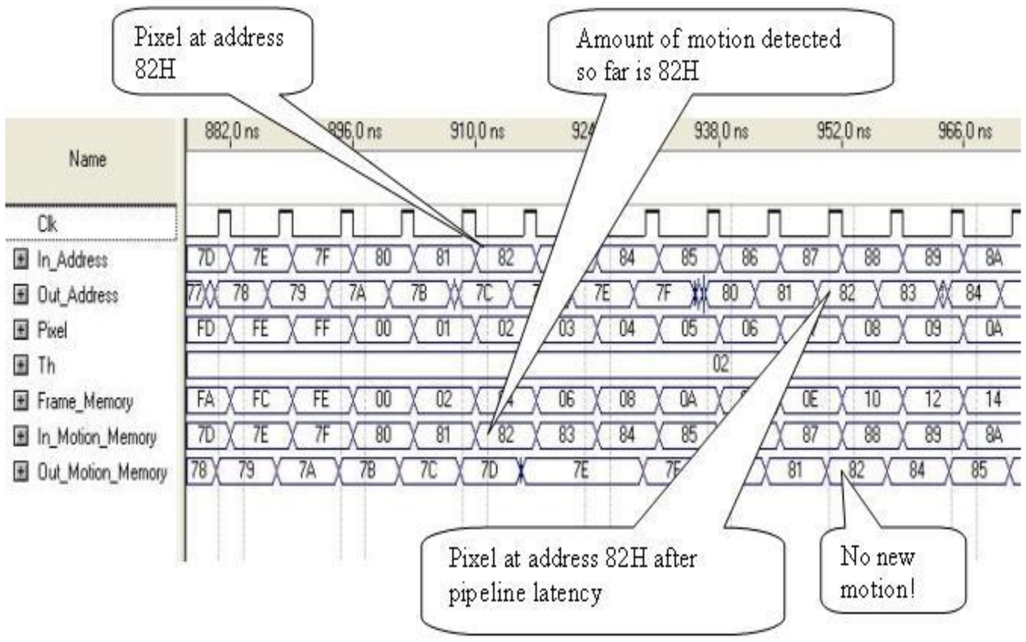


Figure 4: Clouds over the waveform example

7. Grading Policy

Weight	Task	Description
10%	Documentation	The "clear" way in which you presented the requirements and the analysis and conclusions on the work you've done
90%	Analysis and Test	The correct analysis of the system (under the LAB3 SEAT requirements)

Table 3: Grading

Under the above policies you'll be also evaluated using common sense:

- Your files will be compiled and checked, the system must work.
- Your design and architecture must be intelligent, minimal, effective and well organized.