AC19C09

Exemple (! – nu se limitează la acestea) de subiecte teoretice / probleme (partea 1)

1) Masini cu 0, 1, 2, 3 adrese – exemple de program: sa se scrie secvenţa de cod pentru calculul expresiei *E*.... pe maşina cu x adrese, sau de tip stivă, acumulator, …
2) Timpul CPU, performanţă, legea lui Amdahl, etc.
3) Sinteza de nivel înalt HLS: aspecte teoretice plus probleme (exemple in curs)
4) MIPS ISA, instrucţiunile de bază, RTL abstract, pseudo-instrucţiuni
5) Etapele / fazele de proiectare CPU
6) MIPS cu ciclu unic / instrucţiune (Single-cycle MIPS) – căi de date, control, implementarea unor instrucţiuni (exemple concrete urmează)
7) MIPS cu ciclu unic – interfaţare I/O, întreruperi (căi de date, control). Exemplu practic de interfaţare cu un dispozitiv I/O (exemplu I – switchuri, exemplu O - leduri) pe n biti, definire porturi ca adrese de memorie, subliniat pe caile de date modificarile, scris secvente de cod care citesc/scriu pe porturile definite…
8) Detecţia condiţiilor de salt folosind ALU
9) Circuite de deplasare, rotire (Shifters, rotators)
10) Proiectare ALU pentru MIPS
11) Înmulţitoare combinationale
12) Înmulţitor secvenţial cu adunări deplasări repetate
13) Unitate aritmetică reconfigurabilă (intrări/ieşiri pe 8, 16, 32 biti)

16) Proiectarea şi implementarea unei instrucţiuni pe procesorul MIPS single-cycle. Se vor urmării paşii de implementare prin prezentarea: RTL abstract, formatul instrucţiunii, diagrama de procesare, resursele (din căile de date) necesare, valorile semnalelor de control pentru IF şi operaţia asociată, caile de date cu modificarile necesare (daca sunt necesare !).

Exemple de instrucţiuni – vezi cursul

1	Shift Left Logical Variable
2	Shift Left Logical

3	Jump Register
4	Jump and Link Register
5	JM (jump memory). Its instruction format is similar to that of load word except that the rt field is not used because the data loaded from memory is put in the PC instead of the target register
6	SWAP two registers
7	Lwnew Addressing mode (load word) instruction, which sums two registers to obtain the memory address of the data to be loaded and uses the R-format.
8	Load/Store – no offset, only register addressing
9	Arithmetic with memory operand Addmx $t2, $t3 # $t2 = $t2 + Memory[$t3]
10	WAI (where am I), puts the instruction's location (the value of the PC when the instruction was fetched) into a register specified by the rt field of the machine language instruction.
11	Branch on equal zero and link
12	Post increment addressing mode lw $t0, ($t1)+
13	Pre decrement addressing mode sw $t0, -($t1)
14	Push/Pop stack growths upside, SP prepared for POP
15	Push/Pop stack growths downside, SP prepared for PUSH
16	Arithmetic with memory operand Addmy $1,$t2, $t3 # $t1 = $t2 + Memory[$t3]
17	MOV conditional
18	PC relative load
19	IN/OUT - LW/SW
20	SKIPEQ if ($t1 = $t2) → PC ← PC+8

17) Similar cu tipul de probleme de la 16) dar pentru MIPS Multi-ciclu. În plus faţă de cerinţele de la 16) se va prezenta RTL concret pe fazele de execuţie ale instrucţiunii.

18) Similar cu tipul de probleme de la 16) dar pentru MIPS pipeline

1	Shift Left Logical Variable
…	
20	SKIPEQ if ($t1 = $t2) → PC ← PC+8
21	the bcp instruction, copies a block of words from one location in memory to another
22	the block set instruction: bset $t1,$t2,$t3 {mem[t3], mem[t3+1] … mem[t3+t2]} ← $t1
23	Search for a word in a memory block
24	Test a memory block
25	Conditional execution
26	…

19) Unitatea de control de la MIPS Multi-ciclu: diagrama bloc, principiu de funcţionare, formatul de micro-instrucţiune etc.

20) … vezi exemplele de probleme din cursuri (rezolvate / mentionate la sfarsit)