HIGH PERFORMANCE COMPUTER ARCHITECTURE midterm exam 30-06-2008

MATR.NO.____________________

SURNAME____________________

FIRST NAME____________________

1) (34/40) Consider the following fragment of code which is executing on a VLIW processor. Initially R1=630, R2=0x1000, R3=0x3000:

```
lab:   LW     R4, 0(R2)
       LW     R5, 0(R3)
       ADD    R4, R4, R4
       ADD    R4, R4, R5
       MUL    R4, R4          ; R4=R4*R4
       SW     R4, 0(R2)
       SW     R4, 0(R3)
       ADDI   R2, R2, 4
       ADDI   R3, R3, 4
       SUBI   R1, R1, 1
       BNE    R1, R0, lab
```

Working hypothesis:

- Fetch and decode stage have a 5-instruction width
- There are two functional units for the Arithmetic-Logic operations and Branches (ALBUs) with 1 stage merged with the issue stage.
- Branches have 1 delay slot
- There are two Load/Store Units with three stages (effective address calculation, addressing, eventual read); the eventual read requires 1 clock cycle
- Write-backs can be overlapped to the decode stage
- There is one Multiplication Unit (MU) with four stages
- The register file has 24 registers R0-R23 (R0 is hardwired to the value 'zero')
- The register file has 5 independent input ports and 5 independent output ports
- The compiler unrolls the iterations in order to use all available registers (the number of iterations is known by the compiler – initially written in R1)

By compiling the following tables, calculate:

i) the CIT (Cycles per Iteration) of the optimally unrolled loop so that the CIT is minimized;
ii) the IPC (Instructions Per Cycle) at the end of the iterations
iii) the Utilization factor U=available_slots/total_slots

Cycles	ALBU1	ALBU2	LSU1	LSU2	MU	Comments
1	...	...	`LW R4,0(R2)`	`LW R5,0(R3)`	NOP	...
2	...	...	...	...	...	...
3	...	...	...	...	...	...

2) (6/40) Explain the difference between a Superscalar and a VLIW processor: motivate the advantages and disadvantages in each of the two cases.

SOLUTION 2008-06-30

1) The VLIW pipeline is made of the following stages: Fetch (F), Decode (D), Issue (I, eventually subdivided in more stages as in the case of the multiplier → 4 cycles) and Write-Back (W)
Please note that:
   - "Load/Store Units with three stages" implies that the ADD can receive the operand from the LW with a latency of 3 cycles.
   - "Multiplication Unit (MU) with four stages" implies that the SW can receive the result of the MUL with a latency of 4 cycles.
   - "Arithmetic-Logic operations and Branches (ALBUs) with 1 stage merged with the issue stage" implies that the MUL can receive the result of the ADD with no delay.
   - Similarly the BNE will receive the result from the SUBI with no delay.

Cycle	ALBU1	ALBU2	LSU1	LSU2	MU	Comments
1	`NOP`	`NOP`	`LW R4,0(R2)`	`LW R5,0(R3)`	`NOP`	...
2	`NOP`	`NOP`	`LW R6,4(R2)`	`LW R7,4(R3)`	`NOP`	...
3	`NOP`	`NOP`	`LW R8,8(R2)`	`LW R9,8(R3)`	`NOP`	...
4	`NOP`	`NOP`	`LW R10,12(R2)`	`LW R11,12(R3)`	`NOP`	...
5	`ADD R4,R4,R4`	`NOP`	`LW R12,16(R2)`	`LW R13,16(R3)`	`NOP`	...
6	`ADD R4,R4,R5`	`ADD R6,R6,R6`	`LW R14,20(R2)`	`LW R15,20(R3)`	`NOP`	...
7	`ADD R8,R8,R8`	`ADD R6,R6,R7`	`LW R16,24(R2)`	`LW R17,24(R3)`	`MUL R4,R4`	...
8	`ADD R8,R8,R9`	`ADD R10,R10,R10`	`LW R18,28(R2)`	`LW R19,28(R3)`	`MUL R6,R6`	...
9	`ADD R12,R12,R12`	`ADD R10,R10,R11`	`LW R20,32(R2)`	`LW R21,32(R3)`	`MUL R8,R8`	...
10	`ADD R12,R12,R13`	`ADD R14,R14,R14`	`LW R22,36(R2)`	`LW R23,36(R3)`	`MUL R10,R10`	...
11	`ADD R16,R16,R16`	`ADD R14,R14,R15`	`NOP`	`NOP`	`MUL R12,R12`	
12	`ADD R16,R16,R17`	`ADD R18,R18,R18`	`SW R4,0(R2)`	`SW R4,0(R3)`	`MUL R14,R14`	
13	`ADD R20,R20,R20`	`ADD R18,R18,R19`	`SW R6,4(R2)`	`SW R6,4(R3)`	`MUL R16,R16`	
14	`ADD R20,R20,R21`	`ADD R22,R22,R22`	`SW R8,8(R2)`	`SW R8,8(R3)`	`MUL R18,R18`	
15	`NOP`	`ADD R22,R22,R23`	`SW R10,12(R2)`	`SW R10,12(R3)`	`MUL R20,R20`	
16	`NOP`	`NOP`	`SW R12,16(R2)`	`SW R12,16(R3)`	`MUL R22,R22`	...
17	`NOP`	`NOP`	`SW R14,20(R2)`	`SW R14,20(R3)`	`NOP`	...
18	`NOP`	`NOP`	`SW R16,24(R2)`	`SW R16,24(R3)`	`NOP`	...
19	`SUBI R1,R1,10`	`NOP`	`SW R18,28(R2)`	`SW R18,28(R3)`	`NOP`	...
20	`BNE R1,R0,ETIC`	`NOP 0`	`SW R20,32(R2)`	`SW R20,32(R3)`	`NOP`	...
21	`ADDI R2,R2,40`	`ADDI R3,R3,40`	`SW R22,36(R2)`	`SW R22,36(R3)`	`NOP`	...

The new iteration executes 7 instructions for each corresponding instruction in the old iteration (that required 11 instructions) for an unrolling factor of 10. The last two instructions of the old loop need to be modified so that the new iterations proceed regularly (SUB R1,R1,10 and BNE are appropriately adjusted to exploit the delay slot). Moreover, the R2 and R3 pointers must be appropriately adjusted so that the load/store access happens at the right address. Load/Store offset are also statically adjusted. We do not need to take special care of the last iterations since the unrolling factor is a multiple of the statically know number of iterations. The new number of iterations is 630/10=63.

The new iteration executes in 21 cycles 10 of the old iterations, therefore CIT=21/10=2.1
The number of instructions per iteration is 7x10+4=74 instructions in 21 cycles, therefore IPC = 74/21 ≅ 3.52 .
The utilization factor is U = 74/105 ≅ 70%