Computer Architectures
Exam of 04/02/2022

A

First name, Last name, ID: .....

Question 1 (4 points)

Let consider the Branch Prediction Mechanism based on the Branch Target Buffer (BTB).
You are requested to

1. Describe the architecture of a BTB, detailing the modules composing it
2. Describe the behavior of a BTB: when it is accessed, which input information items receives, and which output data it produces each time
3. Assuming that the processor uses 32 bit addresses, each instruction is 4 byte wide, and the BTB is composed of 16 entries, report a table showing the final content of each entry of the BTB and the prediction done for each of the 4 branch instructions if
    - The BTB is initially empty (i.e., full of 0s)
    - The following instructions are executed in sequence
        - `add.d f1, f2, f3` located at the address 0x00022020
        - beq r2, r5, l1 located at the address 0x00022024; the branch is taken, and the branch target address is 0x00022040
        - `addi r1, r0, #1` located at the address 0x00022040
        - `bnez r3,l2` located at the address 0x00022044; the branch is taken, and the branch target address is 0x00022124
        - `addi r1, r0, #2` located at the address 0x00022124
        - `add r2, r0, r5` located at the address 0x00022128
        - `bez r2,l3` located at the address 0x0002212C; the branch is taken, and the branch target address is 0x00022404
        - `bez r2,l4` located at the address 0x00022404; the branch is not taken
        - `add r2, r1, r5` located at the address 0x00022408.

Write your answer here.

1-BTB is composed by N entry, each of M*2 bits where M is the number of bits of an instruction address. First part is the instruction address of the jump, the second part is for the target address of the branch.
2-When an instruction is fetched, the mechanism search for correspondence among entries in the BTB instruction address field, using the least log N significant bits, ignoring the first bits for the instruction alignment. In case of correspondence, the target address is used as prediction. If the prediction was wrong, it will be removed. If a new branch is taken, a new entry in the BTB is allocated.
3-entry 9 -> 0x00022024 0x00022040
Entry 11 -> 0x0002212C 0x00022404
All predictions wrong

Computer Architectures
Exam of 04/02/2022

A

Question 2 (4 points)

Let consider a superscalar MIPS64 architecture implementing dynamic scheduling, speculation and multiple issue and composed of the following units:

- An issue unit able to process 2 instructions per clock period; in the case of a branch instruction only one instruction is issued per clock period
- A commit unit able to process 1 instruction per clock period
- The following functional units (for each unit the number of clock periods to complete one instruction is reported):
  - 1 unit for memory access:1 clock period
  - 1 unit for integer arithmetic instructions: 1 clock period
  - 1 unit for branch instructions: 1 clock period
  - 1 unit for FP multiplication (pipelined): 4 clock periods
  - 1 unit for FP division (unpipelined): 8 clock periods
  - 1 unit for other FP instructions (pipelined): 2 clock periods
- 1 Common Data Bus.

Let also assume that

- Branch predictions are always correct
- All memory accesses never trigger a cache miss.

You should use the following table to describe the behavior of the processor during the execution of the first 2 iterations of a cycle composed of the following instructions, computing the total number of required clock cycles. Register f10 stores a constant.

# iteration		Issue	EXE	MEM	CDB	COMMIT
1	l.d f1,v1(r1)	1	2m	3	4	5
1	l.d f2,v2(r1)	1	3m	4	5	6
1	mul.d f3, f2, f10	2	6mul		10	11
1	add.d f5, f1, f3	2	11a		13	14
1	sub.d f6, f1, f2	3	6a		8	15
1	div.d f7,f5,f6	3	14div		22	23
1	s.d f7,v3(r1)	4	5m			24
1	daddui r1,r1,8	4	5i		6	25
1	daddi r2,r2,-1	5	6i		7	26
1	bnez r2,loop	6	8jmp			27
2	l.d f1,v1(r1)	7	8m	9	11	28
2	l.d f2,v2(r1)	7	9m	10	12	29
2	mul.d f4, f2, f2	8	13mul		17	30
2	add.d f5, f1, f4	8	18a		20	31
2	sub.d f6, f1, f2	9	13a		15	32
2	div.d f7,f5,f6	9	22div		30	33
2	s.d f7,v3(r1)	10	11m			34
2	daddui r1,r1,8	10	11i		14	35
2	daddi r2,r2,-1	11	12i		16	36
2	bnez r2,loop	12	17jmp			37

Computer Architectures
Exam of 04/02/2022

A

Question 3 (4 points)

Given the 4 x 4 matrix SOURCE of words storing only positive data (represented in pure binary on 16 bits), write an 8086 assembly program, which computes the entries of another 4 x 4 matrix MAPP of bytes, according to the following very simple rule:

If SOURCE (i, j) can be represented on 8 bits only, then MAPP (i,j) = 1

Otherwise MAPP (i,j) = 0

- The same program should also compute and store to the variable CROSS (on 16 bits) the sum of all the elements of SOURCE whose corresponding entry in MAPP is equal to 1.
- In your solution, please provide the declaration of SOURCE, MAPP, and CROSS and the code, together with significant comments to the code and instructions.
- Indeed, the choice is yours about how to store the matrices in the memory.

If you have time, in order to get up to one additional point, please also clearly and shortly respond to the following questions (as "comments in the program"): do we risk an overflow for CROSS? Why? Please consider that a wrong response to these optional questions will imply a negative score up to -1.

Write your code in a file saved in the 8086 folder.

Example:

Initial matrix SOURCE

10	20	100	10000
0	7000	1	2
9000	12345	999	30000
200	210	7	65000

Computed MAPP

1	1	1	0
1	0	1	1
0	0	0	0
1	1	1	0

Computed CROSS = 10 + 20 + 100 + 0 + 1 + 2 + 200 + 210 + 7 = 550

Click on the following link to open a web page with the 8086 instruction set:

http://www.jegerlehner.ch/intel/IntelCodeTable.pdf

Computer Architectures
Exam of 04/02/2022

A

Question 4 (8 points)

In the fixed-point representation, a fixed number of digits is used to represent the fractional part of a number. Example of fixed-point number with 8 fractional digits: 101.10011101. The corresponding decimal value is $1 * 2^2 + 0 * 2^1 + 1 * 2^0 + 1 * 2^{-1} + 0 * 2^{-2} + 0 * 2^{-3} + 1 * 2^{-4} + 1 * 2^{-5} + 1 * 2^{-6} + 0 * 2^{-7} + 1 * 2^{-8} = 4 + 0 + 1 + 0.5 + 0 + 0 + 0.0625 + 0.03125 + 0.015625 + 0 + 0.00390625 = 5.61328125$

Write the `restoringSquareRoot` subroutine in ARM assembly language, which returns the square root of a fixed-point number lower than 1. The subroutine receives in input:

- a 32-bit value X
- the number of fractional digits k

The subroutine computes the square root Q following the "restoring" algorithm:

1. Initialization: $r = 2 * X$, $Q = 0$, $T = 2^{-1}$
2. for i = 1 to k:
3. if r >= T:
4. $r = r - T$
5. $Q = Q + 2^{-i}$
6. $T = 2 * Q + 2^{-(i + 1)}$
7. $r = 2 * r$
8. Return Q

Example: X = 0.100110, k = 6
The '.' separating integer and fractional digits is shown only for the sake of clarity.
Initialization: r = 1.001100, Q = 0.000000, T = 0.100000
Iteration 1: r >= T
r = 1.001100 – 0.100000 = 0.101100
Q = 0.000000 + 0.100000 = 0.100000
T = 1.000000 + 0.010000 = 1.010000
r = 2 * 0.101100 = 1.011000
Iteration 2: r >= T
r = 1.011000 – 1.010000 = 0.001000
Q = 0.100000 + 0.010000 = 0.110000
T = 1.100000 + 0.001000 = 1.101000
r = 2 * 0.001000 = 0.010000
Iteration 3: r < T
T = 1.100000 + 0.000100 = 1.100100
r = 2 * 0.010000 = 0.100000
Iteration 4: r < T
T = 1.100000 + 0.000010 = 1.100010
r = 2 * 0.100000 = 1.000000
Iteration 5: r < T
T = 1.100000 + 0.000001 = 1.100001
r = 2 * 1.000000 = 10.000000
Iteration 6: r >= T
r = 10.000000 – 1.100001 = 0.011111
Q = 0.110000 + 0.000001 = 0.110001
T = 1.100010 + 0.000000 = 1.100010
r = 2 * 0.011111 = 0.111110
The subroutine returns Q = 0110001.

Computer Architectures
Exam of 04/02/2022

A

Important notes:

1. **Create a new project with Keil inside the "template" directory and write your code there. The "template" directory contains the subdirectories "led" and "button" that you can add to your project if you need them.**
2. The assembly subroutine must comply with the ARM Architecture Procedure Call Standard (AAPCS) standard (about parameter passing, returned value, callee-saved registers).
3. Click on the following links to open web pages with the ARM instruction set

http://www.keil.com/support/man/docs/armasm

https://developer.arm.com/documentation/ddi0337/e/Introduction/Instruction-set-summary?lang=en

4. You can convert fixed-point numbers from/to base 2 and 10 at this link:

https://www.exploringbinary.com/binary-converter/

Computer Architectures
Exam of 04/02/2022

A

Question 5 (5 points)

Add the following functionalities to the project created in the previous exercise:

1) The user can specify a binary value through buttons Key1 and Key2. Key1 inserts a new digit equal to 0, while key2 inserts a new digit equal to 1. For example, if the user presses Key2, Key1, Key1, Key2, Key2, Key1, the final value is 100110
2) When the user presses INT0, call the `restoringSquareRoot` subroutine passing:
   - the value introduced before with Key1 and Key2 (e.g., 100110)
   - the number of times Key1 and Key2 were pressed before (e.g., 6)

Then, the lowest 8 bits of the value returned by the `restoringSquareRoot` subroutine are shown through the leds 4-11. Led 11 corresponds to the least significant bit. In the example, the lowest 8 bits of the returned value are 00110001, so the led status is:
led 4 off, led 5 off, led 6 on, led 7 on, led 8 off, led 9 off, led 10 off, led 11 on.

The management of the button bouncing is optional (i.e., not required).