Computer architectures

Exam 03/02/2021

	YASSER HOBBALLAH 286584
Iniziato	mercoledì, 3 febbraio 2021, 14:42
Terminato	mercoledì, 3 febbraio 2021, 16:41
Tempo impiegato	1 ora 59 min.

Domanda 1

Completo

Punteggio max.: 4

Let focus on the Reorder Buffer (ROB) existing in the architecture of some superscalar processors.

You are requested to

1. Explain how the ROB works (when an entry is allocated in it, when it is written, when it is read, when it is de-allocated)
2. Describe the ROB architecture, detailing the fields composing each entry
3. Summarize the advantages stemming from the adoption of the ROB.

In the Reorder Buffer architecture, when the issue unit issue an instructions the instruction go to reservation station and reorder buffer only if there is place reservation station for that instruction (free slot) and also if the ROB has place free. otherwise a stall is generated. an entry is alocated when an instruction is issued and it is written from the common data bus, since the CDB is connected directly to the ROB and when an element is written in the CDB the ROB is able to read the operand immediately since it stores for missing operands identifier for the instructions that will issue them in the CDB. The instructions in the ROB can be completed or commited and this is why the ROB can offer speculation where execution of instructions follwoing a branch instruction can excute even if a branch instruction result is not known. completed instructions canot commit (but can complete excution) unless all pervious branches preceding the instruction commit. fields of the are one feld for current result of instruction (commit or compelete or waiting for operands) another field is for alias of missing operands. and a field for the instruction with its number in the ROB to preserve order of instructions since the commit is done in an inorder way (not out of order). the main advantage is that the ROB (implementing speculation) helps in increasing effiency when dealing with branches where instructions following a branch can even excute and this makes the dynamic resduling better in handling branches since without ROB the instructions following a branch will only be issue but not excuted.

Domanda 2

Completo

Punteggio max.: 4

Let consider a MIPS64 architecture including the following functional units (for each unit the number of clock periods to complete one instruction is reported):

- Integer ALU: 1 clock period
- Data memory: 1 clock period
- FP arithmetic unit: 2 clock periods (pipelined)
- FP multiplier unit: 6 clock periods (pipelined)
- FP divider unit: 8 clock periods (unpipelined)

You should also assume that

- The branch delay slot corresponds to 1 clock cycle, and the branch delay slot is not enabled
- Data forwarding is enabled
- The EXE phase can be completed out-of-order.

You should consider the following code fragment and, filling the following tables, determine the pipeline behavior in each clock period, as well as the total number of clock periods required to execute the fragment. The value of the constant k is written in f10 before the beginning of the code fragment.

```
; ********************* MIPS64 ***********************
;  for (i = 0; i < 10; i++) {
;      v4[i] = (v1[i]+v2[i])/(v3[i]*k);
;  }
```

Code	Comments	Clock cycles
.data		
v1: .double "10 values"		

Code		Comments	Clock cycles
v2: .double "10 values"			
v3: .double "10 values"			
.text			
main:	daddui r1, r0, 0	r1 <- pointer	5
	daddui r2,r0,10	r2 <= 10	1
loop:	l.d f1,v1(r1)	f1 <= v1[i]	1
	l.d f2,v2(r1)	f2 <= v2[i]	1
	add.d f5, f1, f2	f5 <= v1[i] +v2[i]	3
	l.d f3,v3(r1)	f3 <= v3[i]	1
	mul.d f6, f3, f10	f6 <= f3*k	7
	div.d f4, f5, f6	f4 <= (v1[i]+v2[i]) /(v3[i]*k)	8
	s.d f4,v4(r1)	v4[i] <= f4	1
	daddui r1,r1,8	r1 <= r1 + 8	1
	daddi r2,r2,-1	r2 <= r2 - 1	1
	bnez r2,loop		2
	halt		1
	total	6+27*10	276

main: daddui r1,r0,0	F	D	E	M	W																											
daddui r2,r0,10		F	D	E	M	W																										
loop: l.d f1,v1(r1)			F	D	E	M	W																									
l.d f2,v2(r1)				F	D	E	M	W																								
add.d f5, f1, f2					F	D	S	E	E	M	W																					
l.d f3,v3(r1)						F	S	D	E	S	M	W																				
mul.d f6, f3, f10								F	D	S	S	E	E	E	E	E	E	M	W													
div.d f4, f5, f6									F	S	S	D	S	S	S	S	S	E	E	E	E	E	E	E	E	M	W					
s.d f4,v4(r1)												F						D	E	S	S	S	S	S	S	S	M	W				
daddui r1,r1,8																		F	D	S	S	S	S	S	S	S	E	M	W			
daddi r2,r2,-1																			F	S	S	S	S	S	S	S	D	E	M	W		
bnez r2,loop																											F	S	D	E	M	W
halt																													F	D	E	M

Domanda 3

Completo

Punteggio max.: 6

Given a 4 x 4 matrix of WORD (i.e. 16 bits single data) SOURCE write a 8086 assembly program which rotates the rows of SOURCE from up to down by 1<=n<=3 positions and stores the result in the matrix DESTINATION, with n given by the user. The choice is yours about how to store the matrices in the memory. Please add significant comments to the code and instructions. If you have time, in order to get one additional point, provide the instructions to extend the program to consider n in the range -80<=n<=+120

Example:

Initial matrix SOURCE

A B C D

E F G H

I J K L

M N O P

if n=3 DESTINATION becomes

E F G H

I J K L

M N O P

A B C D

if n=1 DESTINATION becomes

M N O P

A B C D

E F G H

I J K L

```
.model small
.stack
.data
destination dw 50 dup(0)
source  Dw   'A','B','C','D'
          Dw   'E','F','G','H'
           Dw    'I','J','K','L'
           Dw     'M','N','O','P'
temp dw 0

.code
rotate      proc
    push ax
    push bx
   push cx
push dx
mov temp,26 ;store in temp the position of first element in last row
mov si,26 ; pointer to the first element of last row
mov cx,3  ; external loop counter

loopext:
    push cx
    mov cx,4     ;internal loop counter

loopint:
    mov bx,source[si]   ;get the two elements in position
mov dx,source[si-8]
mov destination[si],dx ;store the two swapped elements in destination matrix
mov destination[si-8],bx

sub si,8 ; go to the next row upperward
dec cx
cmp cx,0
jxxx loopint

pop cx
add temp,2 ; go to the next element in last row
mov si,temp ;initialize pointer si with that element
dec cx  ; decrement counter of outer loop
```

```
cmp cx,0
jxxx loopext ;


pop dx
pop cx
pop bx
pop ax



ret
endp
.startup
xor dx,dx
mov ah,01h
int 21h
CMP al,'1' ; if 1 then rotate matrix once
jxxx rotate1
cmp al,'2'   ; if 2 then rotate twice
jxxx rotate2
cmp al,'3' ; if 3 then rotate 3 times
jxxx rotate3
jxxx finish ; got finish if n is other than 1 or 2 or 3

rotate1:
call rotate
jxxx finish
rotate2:
call rotate
call rotate
jxxx finish
rotate3:
call rotate
call rotate
call rotate
jxxx finish


; this whole section is commented for an alternative solution for n times rotations
; for the additional point i didn't notice the condition and time ran out
; al contains the number n
;cmp al,0
;jxxx rotatentimes1 ; jump to rotatentimes1 if al is positive
;jxxx rotatentimes2 ; al is negative


;rotatentimes1:
;mov cl,al
;call rotate ; call rotate n times
;dec cx
;cmp cx,0
;jxxx rotatentimes1
;jxxx finish

;rotatentimes2:
;mov cl,al
;SUB 0,cx



finish:

.end
exit
```

Informazione

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

Note: Assembly subroutines must comply with the ARM Architecture Procedure Call Standard (AAPCS) standard (about parameter passing, returned value, callee-saved registers).

Domanda 4

Completo

Punteggio max.: 9

The IEEE-754 SP standard expresses floating-point numbers in 32 bits:

31	30 23	22 0
sign	exponent	mantissa

Bit 31 is 0 if the number is positive, 1 if negative.

Write the addFPpositiveNumbers subroutine, which receives in input two 32-bit numbers, considers them as IEEE-754 SP floating point numbers, and returns their sum (in the same format). Bit 31 of the two input numbers is always 0 (i.e., the two numbers are positive).

In details, the subroutine implements the following steps:

1. take the mantissa of the two parameters
2. set the 23rd bit of the mantissa to 1
3. compare the two exponents. If they are equal, the exponent of the result is the same. If they are different:
   - the exponent of the result is the highest one
   - shift right the mantissa of the number with the lower exponent by as many position as the difference between the two exponents.
4. sum the two mantissas: this is the mantissa of the result. If the 24st bit of the mantissa of the result is 1:
   - shift right the mantissa of the result by one position
   - increment the exponent of the result by one.
5. set the 23rd bit of the mantissa of the result to 0.
6. combine the mantissa and the exponent to get the final result.

Example

parameter1 = 0100 0010 0100 1011 0000 0000 0000 0000

parameter2 = 0100 0001 1010 0100 0000 0000 0000 0000

1. mantissa1 = 0000 0000 0100 1011 0000 0000 0000 0000
   mantissa2 = 0000 0000 0010 0100 0000 0000 0000 0000
2. mantissa1 = 0000 0000 1100 1011 0000 0000 0000 0000
   mantissa2 = 0000 0000 1010 0100 0000 0000 0000 0000
3. exponent1 = 1000 0100
   exponent2 = 1000 0011
   - exponentResult = 1000 0100
   - mantissa2 = 0000 0000 0101 0010 0000 0000 0000 0000
4. mantissaResult = 0000 0001 0001 1101 0000 0000 0000 0000
   - mantissaResult = 0000 0000 1000 1110 1000 0000 0000 0000
   - exponentResult = 1000 0101
5. mantissaResult = 0000 0000 0000 1110 1000 0000 0000 0000
6. result = 0100 0010 1000 1110 1000 0000 0000 0000

```
addFPpositiveNumbers  proc
            push{r4,r5,r6,r7,r8,r9,r10,r11,LR}
            ; the two parameters are passed by r0 and r1
            LDR r6,=0x00FFFFFF

            AND r4,r0,r6 ;r4 contains mantiessa of first parameter

            AND r5,r1,r6   ; r5 contain manitessa of second parameter
            LDR r6,=0x00100000
            ORR r4,r6  ;  set the 23rd bit to 1
            ORR r5,r6
            LDR r6,=0x7FF00000
```

```
          AND   r7,r0,r6   ;extract the exponent of the two numbers
          AND   r8,r1,r6

          CMP r7,r8   ;see if the 2 exponents equal
          BEQ cont ; if equal exponent then r7 contains the exponenet
          CMP r7,r8
          BHI expo1 ;if r7 greater than r8 then parameter 1 exponent is higher
          B expo2   ;parameter2 exponenet higher

expo1      SUB r9,r7,r8          ;r7 has higher exponent and r9 contain the difference in exponents
          LSR  r5,r9           ; r5 contains mantiessa of lower number then shift it r9 postions to right



         B cont
expo2      SUB r9,r8,r7          ; r8 has higher exponent
           LSR r4,r9             ; r4 contains mantiessa of lower exponent parameter then shift it to the right by r9 positions


cont                 ; the higher exponent is stored in r7

         ADD r4,r4,r5 ; result of sum of mantiessa is stored in r4
        LDR r6,=0x00800000
        AND r9,r4,r6               ; see if 24th bit of mantiessa is 1
        CMP r9,#0
        BNE  notone
         LSR r4,r4,#1   ; shift postion of result mantiessa
        LSR  r7,r7,#23 ;shift the position of the exponent to become a number
          ADD r7,r7,#1

notone     LDR r6,=0x00800000
          BIC r4,r6  ;set 23rd bit of result mantiessa to 0



          LSL r7,r7,#23  ;shift by 23 postions the position of exponent
          ADD r0,r7,r4   ;add the mantiessa to the exponent and the result is stored in r0



         pop{r4,r5,r6,r7,r8,r9,r10,r11,PC}
endp
```

Domanda 5

Completo

Punteggio max.: 4

Initialize register r4 and r5 with two values expressing floating-point numbers according to the IEEE-754 SP standard.

Configure the SYSTICK timer in order to generate an interrupt every 2^{20} clock cycles.

Enter in an infinite loop. In the SYSTICK timer interrupt handler, sum the content of the r4 and r5 registers by calling the addFPpositiveNumbers subroutine and then store the result in r4.

The SYSTICK timer is configured by means of the following registers:

- Control and Status Register: size 32 bits, address 0xE000E010
- Reload Value Register: size 24 bits, address 0xE000E014
- Current Value Register: 24 bits, address 0xE000E018

The meaning of the bits in the Control and Status Register is as follows:

- Bit 16 (read-only): it is read as 1 if the counter reaches 0 since last time this register is read; it is cleared to 0 when read or when the current counter value is cleared
- Bit 2 (read/write): if 1, the processor free running clock is used; if 0, an external reference clock is used
- Bit 1 (read/write): if 1, an interrupt is generated when the timer reaches 0; if 0, the interrupt is not generated
- Bit 0 (read/write): if 1, SYSTICK timer is enabled; if 0, SYSTICK timer is disabled.

The Reload Value Register stores the value to reload when the timer reaches 0.

The Current Value Register stores the current value of the timer. Writing any number clears its content.

```
;initialize the registers r4 and r5 with values expressing floating point
LDR r4,=0x01230B12
LDR r5,=0x02005303



            ;  lets configure systick timer
          LDR r0,=0xE000E010  ;first turn off the systick timer
            mov r1,#0
              STR r1,[r0]
                 ; next initialize the reload register
          LDR r0,=0xE000E014  ;put the reload value to 2^20 cycles
            mov r2,#2
            LSL r2,#20  ; implement in r2 value 2^20
          mov r1,r2    ;store number of cycles
           STR r1,[r0]
           ; next make sure the current value register is 0 by writing anything on it
           LDR r0,=0xE000E018
          STR r1,[r0]
          ; finally turn on the timer
              LDR r0,=0xE000E010
           LDR r1,[r0]
            ORR r1,#7
        STR r1,[r0] ; timer is ON

stop                         ;infinite loop until exception from timer comes
                 B stop
;in SYSTICK timer inetrrupt handler

       mov r0,r4   ;send the two parameters to subroutine addFPpositiveNumbers
       mov r1,r5
       BL  addFPpositiveNumbers
        mov r4,r0  ;result will be returned in r0 so move it to r4
        BX LR ;return to the main function
```

Domanda 6

Risposta non data

Non valutata

Here you can write:

- explanations on your answers, if you think that something is not clear
- your interpretation of the question, if you had any doubt about the formulation of the question
- any other comments that you want to let the professors know.

You can leave this space blank if you have no comments.