POLITECNICO DI TORINO

Computer architectures

Exam 23/06/2021

ABDULLAH HASSAN
251425

Iniziato mercoledì, 23 giugno 2021, 14:18
Terminato mercoledì, 23 giugno 2021, 16:18
Tempo impiegato 2 ore

Domanda 1

Completo

Punteggio max.: 4

The Tomasulo architecture for superscalar processors with dynamic scheduling and speculation uses one or more Common Data Busses (CDBs).

You are requested to

1. Explain what the CDB is and where it is placed in the Tomasulo architecture, listing the modules able to write on it, and those reading from it
2. Detail when data are written in the CDB
3. List the advantages and disadvantages possibly stemming from the introduction of multiple instances of the CDB.

Signal that is put on the data bus finds its way to other units through data bus ratherr than going through the regissters,whenever a result is processed,common data bus is used to transfer it to different modules.
Operands appeared on CDB are read by reservation unit and to avoid RAW hazards as all instructions are available in reservation unit , it is executed.
2.There are two steps when data is written in CDB:
a).At availability of inBase register , the adress is computed and written in load/store buffer.
b). Execution takes place at availabilty of memory.
New instruction can not initiate execution until all preeceding instructions in branches are completed.Speculation improves this mechanism.

advantage
Allows multiple units waiting on a result to proceed without waiting to resolve contention for access to egister file read ports.
hazards are the situations that do not let the instruction execute at their desired cycle .
these hazards can be eliminated and detected easily
disadvantages
hardware complexity , bottleneck problem can be faced.

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: 4 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 values of the constants k1 and k2 are written in f10 and f11, respectively, before the beginning of the code fragment.

```
; ********************* MIPS64 ************************
; for (i = 0; i < 100; i++) {
;
        v4[i] = ((v1[i]/k1) + (v2[i]/k2))*v3[i];
; }
```

Code Comments Clock cycles .data

v1: .double "100 values"

v2: .double "100 values"

v3: .double "100 values"

v4: .double "100 values"

.text

main: daddui r1,r0,0 r1← pointer 5

daddui r2,r0,100 r2 <= 100 1

loop: l.d f1,v1(r1) f1 <= v1[i] 1

div.d f2, f1, f10 f2 <= v1[i]/k1 9

l.d f3,v2(r1) f3 <= v2[i] 0

div.d f4, f3, f11 f4 <= v2[i]/k2 8

add.d f5,f2,f4 f6 <= v1[i]/k1+v2[i]/k2 2

l.d f6,v3(r1) f5 <= v3[i] 1

mul.d f7,f5,f6 f7 <= (v1[i]/k1+v2[i]/k2)*v3[i] 5

s.d f7,v4(r1) v4[i] <= f7 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+(3200)=3206

r1,r0,0 F **daddui**

main:
daddui D E M W 5 div=8 mul= 4 al=2

r2,r0,100 F D E M W 1 **loop:**
l.d F D E M W 1
f1,v1(r1) div.d f2,

f1, f10 F D - E E E E E E E E M W 9

l.d

f3,v2(r1) F - D E M W 0

div.d f4,

f3, f11 F D - E E E E E E E E M W

add.d

f5,f2,f4 F - D - - - - - - - E E M W **l.d**

f6,v3(r1) F - D E - M W **mul.d**

f7,f5,f6 - F D - - E E E E M **s.d**

f7,v4(r1) F D E **daddui**

r1,r1,8 F D 6/23/21 4:28 PM Pagina 2 di 7

daddi

r2,r2,-1 F **bnez**

r2,loop - **halt**

Domanda 3

Completo

Punteggio max.: 6

Given a 5 x 5 matrix of bytes SOURCE representing unsigned numbers, write a 8086 assembly program which computes on 16 bits the sum of all cells excluding these on the main diagonal, i.e. upper left-to-lower-right diagonal, minus the sum of all the cells of the same main diagonal.

Please add significant comments to the code and instructions.

Friendly advice: before starting to write down the code, think at a possible (very) simple algorithm! The choice of the algorithm highly influences the complexity and length of the code.

Example:

matrix SOURCE

1 2 3 4 5

6 7 8 9 0

9 8 7 6 5

4 3 2 1 0

7 7 7 7 7

all cells excluding the main diagonal:

2+3+4+5+

6+8+9+0+

9+8+6+5+

4+3+2+0+

7+7+7+7= 102

all cells on the main diagonal

1+

7+

7+

1+

7= 23

Result (on 16 bits in two’s complement) = 102-23 = 79

```
.MODEL SMALL
.STACK
.DATA
MAT DB
1,2,3,4,5,6,7,8,9,0,9,8,7,6,5,4,3,2,1,0,7,7,7,7,7; cutting by row
ARR DB 25 DUP(?)
OUTPUT1 DW 0
OUTPUT2 DW 0
RESULT DW 0
.CODE
.STARTUP

;moving from MAT to ARR
MOV BX,0
```



```
MOV CX,25 ; counter
BLOCK:
MOV DL,MAT[BX]
MOV ARR[BX],DL
INC BX
DEC CX
CMP CX,0
JNZ BLOCK

; adding diagonal element to OUTPUT2 and replacing with 0
MOV CX,5
MOV DX,0                    MOV BX , 0
MOV AX,0
ADDINGDIAGONAL:
MOV AL,ARR[DX]              MOV AL , ARR[ BX]
MOV ARR[DX],0               MOV ARR[ BX] , 0
ADD OUTPUT2,AX
INC DX                      INC BX
DEC CX
ADD DX,5                    ADD BX , 5
CMP CX,0
JNZ ADDINGDIAGONAL

: adding all other elements to OUTPUT1
MOV CX,25
DOV DX,0                    MOV BX , 0
MOV AX,0
ADDINGOTHER:
MOV AL,ARR[DX]              MOV AL , ARR[BX]
ADD OUTPUT1,AX
INC DX                      INC BX
DEC CX
CMP CX,0
JNZ ADDINGOTHER

; substracting and writing to result
MOV AL,OUTPUT1              MOV AX , OUTPUT1
MOV BL,OUTPUT2              MOV BX , OUTPUT2

SUB REULT,AL,BL             SUB AX  , BX

MOV RESULT, AX              MOV RESULT, AX
.EXIT
END
```

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.: 4

Write the getMaxAbsoluteValue function in C language, having the following prototype:

int getMaxAbsoluteValue(float parameter1, float parameter2)

The function returns 1 if the absolute value of parameter1 is higher than or equal to the absolute value of parameter2, 0 otherwise.

Then, write the Reset_Handler procedure in an assembly file that calls the C function passing two parameters.

Note 1: the two parameters in the Reset_Handler can be initialized to any value; anyway, it should be noted that their value is considered according to the IEEE-754 SP standard. This standard expresses floating-point numbers in 32 bits:

31 30 23 22 0

sig n	exponent	mantissa

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

Note 2: it is important to add proper directives in the C and/or assembly file in order to guarantee the visibility of the getMaxAbsoluteValue function.

Content of the C file:

```
#include <math.h>
int getMaxAbsoluteValue(float param1, float param2 ) {
    float a = fabsf(param1);
    float b = fabsf(param2);
    if(a > b){
        return 0;
    }
     return 1;
}
```

Content of the assembly file:

AREA . TEXT

```
CODE READONLY
PARA 1 DCD 0 *10000000
Parameter2 DCD 0*20000000
reset_handler proc
export reset_handler [weak]
                                        EXTERN getMaxAbsoluteValue
                                        MOV r0, #3
                                        MOV r1, #4

LDR r0 = para1
LDR r0[ro]
LDR r1[r1]
BL getmaxAbsoulutevalue
endp
getmaxAbsolutevalue proc
cmp r1 , r0 ; IF R1>0
mov gt r2 , 0 ; IF R1> R2= 0
movle r2 , 1 ; R1 LESS OR EQUALR2=1
pop r2
endp
```


Domanda 5

Completo

Punteggio max.: 9

Write the getAbsoluteDifference subroutine in ARM assembly, which receives in input two 32-bit numbers, considers them as IEEE-754 SP floating point numbers, and returns their absolute difference (in the same format).

In details, the subroutine implements the following steps:

1. pass the two parameters to the getMaxAbsoluteValue function. If the result of the function is 0, swap the two parameters
2. the exponent of the result is the same as the exponent of the first parameter; the sign of the result is 0
3. take the mantissa of the two parameters
4. set bit 23 of both mantissas to 1
5. if the exponent of the second parameter is lower than the exponent of the first parameter, shift right the mantissa of the second parameter by as many positions as the difference between the two exponents
6. check the sign of the two parameters.
   If the sign is the same:
   a) the mantissa of the result is the difference between the mantissa of the first parameter and the mantissa of the second parameter
   b) As long as bit 23 of the mantissa of the result is 0:
   - shift left the mantissa of the result by one position
   - decrement the exponent of the result by one

   Instead, if the two parameters have different sign:
   a) sum the two mantissas: this is the mantissa of the result

b) If bit 24 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

7. set bit 23 of the mantissa of the result to 0
8. combine sign, mantissa and exponent to get the final result

Example:

parameter1 = 0100 0000 0100 1001 0000 1111 1101 1011

parameter2 = 1100 0001 1111 0110 1100 1011 1110 0100

1. the getMaxAbsoluteValue function returns 0, so:
   parameter1 = 1100 0001 1111 0110 1100 1011 1110 0100
   parameter2 = 0100 0000 0100 1001 0000 1111 1101 1011
2. exponentResult = 1000 0011
   signResult = 0
3. mantissa1 = 0000 0000 0111 0110 1100 1011 1110 0100
   mantissa2 = 0000 0000 0100 1001 0000 1111 1101 1011
4. mantissa1 = 0000 0000 1111 0110 1100 1011 1110 0100
   mantissa2 = 0000 0000 1100 1001 0000 1111 1101 1011
5. exponent1 = 1000 0011
   exponent2 = 1000 0000
   mantissa2 = 0000 0000 0001 1001 0010 0001 1111 1011
6. the parameters have different sign
   a) mantissaResult = 0000 0001 0000 1111 1110 1101 1101 1111
   b) mantissaResult = 0000 0000 1000 0111 1111 0110 1110 1111
   exponentResult = 1000 0100
7. mantissaResult = 0000 0000 0000 0111 1111 0110 1110 1111
8. result = 0100 0010 0000 0111 1111 0110 1110 1111

Area . text

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.