CHALMERS

2022-05-18

Tentamen i EDA333 (DIT 122) Datorsystemteknik

Tid: 2 juni, 8:30 - 12:30

Ansvarig för examination: Per Stenström, Tel: 0730-346 340

Tillåtna hjälpmedel: Chalmers godkänd kalkylator, utdelat "green card" med MIPS instruktioner.

Tentamensvisning: Mer information om detta kommer att publiceras på Canvas.

Betygsinterval:

- **Underkänd**: Resultat < 24
- **Betyg 3**: 24 <= Resultat < 36
- **Betyg 4:** 36 <= Resultat < 48
- **Betyg 5:** 48 <= Resultat

Följande gäller:

- Kursbetyg är en sammanvägning av betyg på tentamen och projektuppgift. För godkänt måste båda moment vara godkända. Därutöver kommer slutbetyget att bestämmas genom att bonuspoäng motsvarande 6p och 12p adderas till tentapoängen vid betyg 4 respektive 5 på projektuppgiften.
- Då icke-svensk talande assistenter deltar i tentarättning ber jag om svar på engelska.
- Då flera assistenter är med vid tentarättning får endast en uppgift (1-4) redovisas per blad.

Lycka till!

Per Stenström

CHALMERS UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
412 96 Göteborg
Visiting address: Rännvägen 5
Phone: 031-772 1761 Fax: 031-772 3663
Org. Nr: 556479-5598
E-mail: pers@chalmers.se

[Disclaimer

If you find that the necessary facts are not stated to solve a task, then choose to either 1) ask the teacher during his / her visit or 2) make your own assumptions. These will be approved if they are necessary to solve the task and do not make the task easier. Be sure to always carefully motivate your answers**.]**

Assignment 1

The following code sums two vectors (A and B) and puts the result in a third vector (C).

Vectors A, B and C store 32-bit **floating point** numbers.

```
float A[100], B[100], C[100]
for (i=1; i<100; i++)
    C[i]=A[i]+B[i];
```

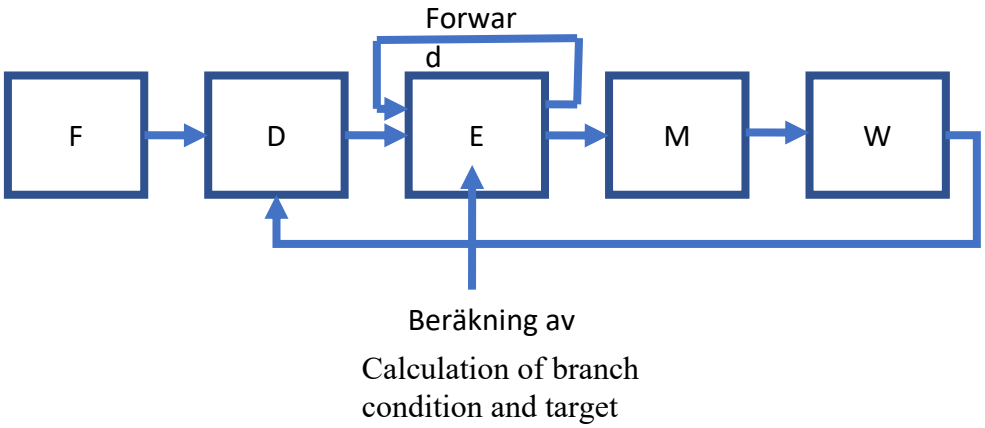
a) Code this high level code in MIPS assembly. The only values found in the registers before execution are the base addresses of A, B and C, in $ a0, $ a1 and $ a2, respectively. Follow the register and call convention for the MIPS processor. Comment on the code. (**6 points)**

b) We have a 32-bit floating-point number with the sign bit 1, the mantissa is 10000 ... 0_2 and the exponent is 01000000_2. The bias for the 32-bit floating point number is 127. Which decimal number corresponds to this floating-point number? (**3 points**)

c) How many instructions are executed and how many times are the instruction and data memory accessed? (**3 points**)

Assignment 2

An implementation of a MIPS computer has a five-step pipeline (F-Fetch, D-Decode, E-Execute, M-Memory Access and W-Write Back) as follows.



Note the following: 1) Branch target and branch conditions are calculated in the execution stage (E). 2) There is forward logic from the output (after the pipeline register) back to the input of the E-stage. 3) However, there is no forward logic from the output of the M-stage (after the pipeline register) back to the E-stage. You can read an operand from the register file only the cycle after it has been written to it.

Consider the following MIPS code:

```
loop:   lw  $t1, 0($a1)
        lw  $t2, 0($a2)
        add  $t3, $t1, $t2
        sw  $t3, 4($a1)
        addi  $a1, 4
        addi  $a2, 4
        addi  $t0, 1
        bne  $t0, $t5, loop
```

a) How many cycles does it take to execute two iterations from the time the first instruction in the first iteration is fetched to the first instruction of the third iteration is fetched? **Tip**: Create a pipeline diagram (X-axis time in clock cycles and the Y-axis instructions in the code as retrieved from the instruction memory) (**6 points**)

b) How many cycles could have been eliminated if forward logic was introduced from the output of the M-stage (after the pipeline register) back to the E-stage? (**3 points**)

c) How many cycles could have been eliminated if we had also been able to calculate the branch target address and branch conditions in the F-step (e.g ., by jump prediction)? (**3 points**)

Assignment 3

A processor is clocked at a clock frequency of 1 GHz. A program running on this processor has three distinct phases. In the first phase, 1 million instructions are executed with a CPI of 1.5 without regard to the impact of cache misses and the proportion of executed load instructions is 20%. In the second phase, 2 million instructions are executed and here CPI is 3 without regard to cache misses and the proportion of executed load instructions is 10%. In the third phase, 1 million instructions are executed and the CPI is 1 without taking into account cache misses and the proportion of executed reading instructions is 20%.

The processor is connected to an instruction and data cache and the following applies:

- The miss rate for the instruction and data cache is 20% and 10%, respectively
- The time to handle misses for the instruction and data cache is in both cases 20 ns.

a) Determine the execution time of the program. (**6 points**)

b) Consider switching processor to a model that has the same instruction set but halves the CPI (excluding the instruction and data cache) but doubles the time to handle cache misses. How much faster / slower is this processor compared to the original? (**6 points**)

c) A completely different processor model is now being considered. This model has an improved cache memory architecture that leads to halving the miss rate for both instructions and data but leads to an extra cycle to access the data cache. Does the new processor lead to higher performance and if so, how much compared to the old one? (**6 points**)

Assignment 4

A computer system has virtual memory with 32-bit virtual addresses and 26-bit physical addresses and a page size of 4 KiB. Each entry in the page table contains 4 bytes. The computer system can run a total of 16 processes simultaneously.

The computer system also contains a two-level cache hierarchy. The first level contains an instruction and data cache and both are 64 KiB and have the block size 16 B. The instruction cache is direct mapped while the data cache is 4-way set associative. The second level cache is shared between the first level instruction and data cache and is 1 MiB. It is 8-way set associative and also has a block size of 16 B. All caches support write-back for writes.

The computer system also contains a TLB where each entry in addition to address translation contains a valid, dirty and a reference bit.

a) Calculate the size of the page table. (**6 points**)

b) Calculate the size of each block entry in the three different caches. (**9 points**)

c) Calculate the size of an entry in the TLB. (**3 points**)

****** Good Luck! ******

Solutions to the exam in EDA333/DIT122 2022-06-02

Assignment 1

a)

```
loop:   lw $f1, 0($a1) # läs in A[i] i f1
        lw $f2, 0($a2) # läs in B[i] i f2
        add.s $f3, $f1, $f2     # put A[i] + B[i] in f3 (flyttal)
        sw  $f3, 4($a3)         # put f3 in C[i]
        addi  $a1, 4            # increment address register for A[i]
        addi  $a2, 4            # increment address register for B[i]
        addi  $a3, 4            # increment address register for C[i]
        addi  $t0, 1            # i=i+1;
        bne  $t0, $t5, loop     # t5 is assumed to contain 100. Branch if i < 100
```

b)
The floating point number contains: S=1, mantissa=1000..000$_2$ and exponent=01000000$_2$

This corresponds to $(-1)^S$ x 1.1000..000 x $2^{64-Bias}$= -1.5 x 2^{64-127}=-1.5x2^{-63}

c) There are 9 instructions in each iteration and 100 iterations are executed. Thus, 900 accesses are made to the instruction memory. In each iteration there are three data memory accesses. Thus, the data memory is accessed 300 times.

Assignment 2

Let's enumerate the instructions as follows:

```
I1:loop:  lw  $t1, 0($a1)
I2:        lw  $t2, 0($a2)
I3:        add  $t3, $t1, $t2
I4:        sw  $t3, 4($a1)
I5:        addi  $a1, 4
I6:        addi  $a2, 4
I7:        addi  $t0, 1
I8:        bne  $t0, $t5, loop
```

Let us first draw up a pipeline diagram for the instructions in iteration 1 where Ii,j corresponds to instruction i in iteration j. X corresponds to a bubble in the pipeline. We can state that instruction I8.1 (the jump instruction) reaches the E step in cycle 12. Thus, the first instruction in the second iteration can be retrieved first in cycle 13.

a)

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14
I1,1	F	D	E	M	W									
I2,1		F	D	E	M	W								
I3,1			F	D	X	X	E	M	W					
I4,1				F	X	X	D	E	M	W				
I5,1							F	D	E	M	W			
I6,1								F	D	E	M	W		
I7,1									F	D	E	M	W	
I8,1										F	D	E	M	W

The next pipeline diagram contains the instructions in the second iteration. The target address from I8,1 is available in cycle C13 when I1,2 will be fetched. We note that branch instruction I8,2 reaches the E stage in cycle 24 and therefore the first instruction in the third iteration can be fetched in cycle 25.

	C13	C 14	C 15	C 16	C 17	C 18	C 19	C 20	C 21	C 22	C 23	C 24	C 25	C 26	C 27
I1,2	F	D	E	M	W										
I2,2		F	D	E	M	W									
I3,2			F	D	X	X	E	M	W						
I4,2				F	X	X	D	E	M	W					
I5,2							F	D	E	M	W				
I6,2								F	D	E	M	W			
I7,2									F	D	E	M	W		
I8,2										F	D	E	M	W	
I1,3													F	D	E

Answer: 25 cycles

b) The number of bubbles for I3,1 is reduced by one. Branch instruction I8,1 has calculated the target address and branch condition in cycle 11.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13
I1,1	F	D	E	M	W								
I2,1		F	D	E	M	W							
I3,1			F	D	X	E	M	W					
I4,1				F	X	D	E	M	W				
I5,1						F	D	E	M	W			
I6,1							F	D	E	M	W		
I7,1								F	D	E	M	W	
I8,1									F	D	E	M	W

Hence, instruction I1,2 can be fetched in cycle 12.

	C12	C 13	C 14	C 15	C 16	C 17	C 18	C 19	C 20	C 21	C 22	C 23	C 24	C 25	C 26
I1,2	F	D	E	M	W										
I2,2		F	D	E	M	W									
I3,2			F	D	X	E	M	W							
I4,2				F	X	D	E	M	W						
I5,2						F	D	E	M	W					
I6,2							F	D	E	M	W				
I7,2								F	D	E	M	W			
I8,2									F	D	E	M	W		
I1,3												F	D		

The first instruction in iteration 3 is thus fetched in cycle 23 (instead of cycle 25)

c)

Now the first instruction in iterations 2 and 3 can be retrieved two cycles earlier, which reduces the number of cycles in total by 4 to 19 cycles.

Assignment 3

a) We use T_{CPU} = IC x CPI x T_c

$CPI_i = CPI_{base,i} + CPI_{cache,i}$ is CPI in phase i.

We start with phase 1:
$CPI_{base,1}$=1.5 and $CPI_{cache,1}$=0.2x20ns/1ns + 0.1x0.2x20ns/1ns=4.4. Hence, CPI_1=5.9
Phase 2:
$CPI_{base,2}$=3 and $CPI_{cache,2}$=0.2x20ns/1ns + 0.1x0.1x20ns/1ns=4.2. Hence, CPI_2=7.2

Phase 3:
$CPI_{base,3}$=1 and $CPI_{cache,3}$=0.2x20ns/1ns + 0.1x0.2x20ns/1ns=4.4. Hence CPI_3=5.4

$T = I_1xCPI_1xT_c + I_2xCPI_2xT_c + I_3xCPI_3xT_c = (1 \times 5.9 + 2 \times 7.2 + 1 \times 5.4)10^6x10^{-9}$ s = **25.7 ms**

b) The new processor model has the same instruction set. This means that it executes the same number of instructions. But CPI for the new model without taking cache misses into account is halved. Hence,

$CPI_{base,1}$=0.75, $CPI_{base,2}$=1.5 and $CPI_{base,3}$=0.5. The time for cache misses doubles. We get:
Phase 1: CPI_1=0.75 + 8.8 = 9.55
Phase 2: CPI_2=1.5 + 8.4 = 9.9
Phase 3: CPI_3=0.5 + 8.8 = 9.3
$T_{ny} = I_1xCPI_1xT_c + I_2xCPI_2xT_c + I_3xCPI_3xT_c = (1 \times 9.55 + 2 \times 9.9 + 1 \times 9.3)10^6x10^{-9}$ s = **38.65 ms**

The new model has lower performance by a factor 38.65/25.7 ~ 1.4

c)
The cache hit time takes one cycle longer. This means:

Phase 1:
$CPI_{base,1}$=1.5 and $CPI_{cache,1}$=0+ 0.1x20ns/1ns + 1x0.2 + 0.05x0.2x20ns/1ns=2.4 dvs CPI_1=3.9

Phase 2:
$CPI_{base,2}$=3 and $CPI_{cache,2}$=0+0.1x20ns/1ns + 0.1x1 + 0.05x0.1x20ns/1ns=2.2 dvs CPI_2=5.2

Phase 3:
$CPI_{base,3}$=1 and $CPI_{cache,3}$=0 + 0.1x20ns/1ns + 1x0.2 + 0.05x0.2x20ns/1ns=2.4 dvs CPI_3=3.4

$T_{ny} = I_1xCPI_1xT_c + I_2xCPI_2xT_c + I_3xCPI_3xT_c = (1 \times 3.9 + 2 \times 5.2 + 1 \times 3.4)10^6x10^{-9}$ s = **17.7 ms**

Hence, this model is 25.7 / 17.7 = 1.45 meaning 45% faster than the original.

Assignment 4

a)

Page table size: The number of virtual pages x the size of each page table entry.

Number of virtual pages = Total virtual address space / page size = number of processes x size of each process virtual address space / page size.

The size of the address space for a single process: 2^{32}
Number of processes: $16 = 2^4$

Size of page table = 2^{32} x 2^4 x 4 / 2^{12} = 2^{26} B = 64 MiB.

b) We start with the instruction cache. There are 64 KiB / 16 B = 4096 blocks in this cache.

The physical address is 26 bits and of these 12 bits are needed for index and 4 bits for offset. There are 26 - 12 - 4 left = 10 bits for the tag. In addition to this, a valid bit is needed for each block and since the cache uses write-back, a dirty bit is also needed.

For each block you need 2 status bits (valid and dirty) + 10 tag bits + 16 x 8 bits for data, i.e., 140 bits.

The data cache: The only difference compared to the instruction cache is that it is a 4-way set associative. This leads to two more tag bits. Thus, each entry becomes **142 bits.**

The second level cache: It contains 1 MiB / 16 B blocks = 65536 blocks. If it were direct mapped, the number of tag bits would have been 26 - 16 - 4 = 6 bits. But since it is an 8-way set associative, 3 tag bits are added. Thus, each entry comprises i 2 status bits + 9 tag bits + 16 x 8 bits = **139 bits**.

c) Each entry in the TLB contains a physical page number and 3 status bits (valid, dirty and reference). A physical page number occupies 13 bits because the page size is 4 KiB, and the physical address is 26 bits, so 26-12=13 bits. Thus, each entry occupies **16 bits**.