Examen computerarchitectuur

Vrijdag 6 juni 2008, 14:00

Prof. Koen De Bosschere

Naam, Voornaam: Richting:

Belangrijk

1. Vergeet niet uw naam en voornaam te vermelden.
2. Schrijf de antwoorden in de daarvoor voorziene ruimte. Bereid uw antwoord voor in het klad, en schrijf het naderhand over. De antwoorden zijn meestal kort.
3. Het examen duurt 3 uur.
4. Gelieve geen rode inkt te gebruiken.
5. Het examen is open boek.
6. U mag geen computer, gsm of rekenmachine gebruiken bij het oplossen van de vragen.
7. Onregelmatigheden worden aan de examencommissie gemeld.

Veel succes!

Ik verklaar op erewoord dat ik noch hulp geboden heb aan, noch hulp ontvangen heb van derden tijdens het oplossen van dit examen.
Handtekening:

Schrijf hier eventuele opmerkingen die van belang kunnen zijn bij de quotering (ziekte, topsport, gemaakte afspraken, enz.).

Vraag 1 (4 punten)

BCD-codering and packed BCD-coderingen gebruiken 8 resp. 4 bits per te coderen decimaal. Naast de besproken BCD-coderingen bestaan er ook ‘Densely Packed BCD’ coderingen. Het idee achter deze coderingen is dat men in plaats van decimaal per decimaal te coderen, groepen van decimalen codeert (typisch 2 of 3) en deze in een minimaal aantal bits codeert. Vergelijk het aantal representeerbare getallen met 32 bits.

	# representeerbare getallen
Binair	
Unpacked BCD	Fractie van binaire voorstelling:
Packed BCD	Fractie van binaire voorstelling:
Densely Packed BCD	Stel jouw eigen DP BCD voorstelling voor. Geef duidelijk aan welke groepjes van decimalen je samen wilt nemen (max per groepje = 5 decimalen, je mag ook combineren). Bereken het aantal representeerbare (volwaardige) decimalen, en de fractie van het aantal getallen dat in de binaire voorstelling kan voorgesteld worden. Fractie van binaire voorstelling:

Vraag 2 (4 punten)

Beschouw het volgende C-programma

```
#include <stdio.h>

int len(char *s)
{
  int n = 0;
  while (*s++) n++;
  return n;
}

int main()
{
  char *input = "computerarchitectuur";

  printf("len(%s) = %d\n", input, len(input));
}
```

Met als code die door de gcc-compiler gegenereerd wordt: (optimalisatieniveau 3).

```
len:    pushl   %ebp
        movl    %esp, %eax
        movl    8(%ebp), %eax
        leal    1(%eax), %edx
        cmpb    $0, (%eax)
        jne     .L2
        xorl    %eax, %eax
        leave
        ret
.L2:    xorl    %eax, %eax
.L5:    incl    %eax
        cmpb    $0, -1(%eax,%edx)
        jne     .L2
        leave
        ret

.LC0:   .string "computerarchitectuur"
.LC1:   .string "len(%s) = %d\n"

main:   pushl   %ebp
        movl    %esp, %ebp
        subl    $8, %esp
        andl    $-16, %esp
        subl    $16, %esp
        movl    $.LC0, %edx
        subl    $1, %edx
.L10:   movb    1(%edx), %al
        incl    %edx
        testb   %al, %al
        jne     .L10
        pushl   %eax
        subl    $.LC0, %edx
        pushl   %edx
        pushl   $.LC0
        pushl   $.LC1
        call    printf
        addl    $16, %esp
        leave
        ret
```

In de functie len werden er 2 fouten geïntroduceerd door 2 instructies te veranderen (geen instructies toegevoegd of weggehaald). Verbeter deze instructies in de code.

In de functie main heeft de compiler een vrij ingrijpende optimalisatie uitgevoerd. Geef aan dewelke.

Teken de stapel op het ogenblik dat de functie printf opgeroepen wordt. Je mag veronderstellen dat %esp initieel op 00FF FF0C staat.

Vraag 3 (4 punten)

In de cursus werden de 4-bit doorsijpelopteller en de 4-bit Carry Lookahead Opteller besproken, inclusief het aantal poortvertragingen per uitgang. Als we nu 4 4-bit CLA's in cascade schakelen, wat zullen de poortvertragingen voor de 16 uitgangen + carry-out dan zijn?

bit	Doorsijpelopteller	CLA
c-in	0	0
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
c-out		

Vraag 4a (2 punten)

Gegeven de volgende instructiemix:

Instructie	Freq.
Controletransfers	23%
ALU-instructies	47%
Leesoperaties	17%
Schrijfoperaties	13%

Vergelijk de prestatie (in klokcycli per instructie, CPI)

van een architectuur met een vertraagde controletransfer met één delay slot waar in 50% van de gevallen een instructie kan gevonden worden die het delay slot op een zinvolle manier kan invullen (A1),

met een architectuur met een sprongvoorspeller waar 90% van alle controletransfers perfect voorspeld kunnen worden, en er voor de resterende controletransfers 3 extra cycli vereist zijn (A2)

Je mag ervan uitgaan dat alle andere instructies een CPI van 1 hebben, en dat de aangebrachte wijzigingen geen secundaire effecten op de rest van de uitvoering hebben.

CPI(A1) =

CPI(A2) =

Vraag 4b (2 punten)

Beschouw de volgende code (Escape syntax):

```
32:          ADDI R0, 0x0005, R3
36: lab2     ADDI R0, 0x0005, R1
40: lab1     ADD  R2, R2, R2
44:          SUBI R1, 0x0001, R1
48:          BRGT R1, lab1
52:          SUBI R3, 0x0001, R3
56:          BRGT R3, lab2
60:          ADD  R9, R10, R11
```

Index	Waarde
0	1
1	0
2	1
3	0
4	1
5	1
6	0
7	1

De tabel bevat de initiële inhoud van een eenvoudige dynamische sprongvoorspeller. De functie die het adres van de instructie afbeeldt op de tabel is ‘direct mapped’. Vraag: hoeveel spronginstructies zullen er door deze code in totaal uitgevoerd worden, en hoeveel daarvan zullen er juist voorspeld worden? Hoe zal het aantal juist voorspelde sprongen evolueren indien er een 2-bit teller gebruikt wordt (0 wordt dan 00 en 1 wordt dan 11 in de initiële tabel; de teller telt gewoon op en af).

sprongen =

correct voorspelde sprongen bij 1 bit voorspeller =

correct voorspelde sprongen bij 2 bit voorspeller =

Vraag 5 (4 punten)

Gegeven een 4-wegs set-associatieve cache met de volgende inhoud.

IDX	Tag	V	Bytes				Tag	V	Bytes				Tag	V	Bytes				Tag	V	Bytes			
0	84	1	ED	32	0A	A2	9E	0	BF	80	1D	FC	10	0	EF	09	86	2A	E8	0	25	44	6F	1A
1	18	1	03	3E	CD	38	E4	0	16	7B	ED	5A	84	0	8E	4C	DF	18	E6	0	FB	B7	12	02
2	80	0	54	9E	1E	FA	84	1	DC	81	B2	14	48	0	B6	1F	7B	44	89	1	10	F5	B8	2E
3	92	0	2F	7E	3D	A8	9F	0	27	95	A4	74	57	1	07	11	FF	D8	93	1	C7	B7	AF	C2
4	84	0	32	21	1C	2C	FA	1	22	C2	DC	34	73	0	BA	DD	37	D8	28	1	E7	A2	39	BA
5	A7	1	A9	76	2B	EE	73	0	BC	91	D5	92	28	1	80	BA	9B	F6	84	1	48	16	81	0A
6	AA	0	5D	4D	F7	DA	29	1	69	C2	8C	74	B5	1	A8	CE	7F	DA	AA	1	FA	93	EB	48
7	84	1	04	2A	32	6A	96	0	B1	86	56	0E	CC	0	96	30	47	F2	91	1	F8	1D	42	30

Het geheugen is byte adresseerbaar, de adressen zijn 13 bit breed, de cache haalt blokken van 4 bytes op uit het geheugen. De kolom V bevat het valid bit, de kolom IDX bevat de index.

Gegeven een adres van 13 bits. Geef voor elk bit aan of het gebruikt zal worden als Index bit (I), tag bit (T) of offset bit (O).

12	11	10	9	8	7	6	5	4	3	2	1	0

Geef voor elk van de volgende adressen aan of een leesoperatie van 1 byte op het aangegeven adres een cachetreffer (T) of -misser (M) zal veroorzaken, en welke byte zal teruggegeven worden in het geval van een cachetreffer.

Adres	T/M	byte
0AEE		
0D74		
155A		
1C87		

Wat zal de hitrate zijn indien alle bytes uit het bereik 1080-109F éénmaal moeten gelezen worden?