Examen computerarchitectuur

Dinsdag 16 juni 2009, 14u00

Prof. Koen De Bosschere

Richting: Naam:

Belangrijk

1. Vergeet niet uw naam 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 de oplossing van de vragen.
7. Gelieve uw mobieltje uit te schakelen.
8. 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)

Gegeven de volgende twee vlottende-kommagetallen in 1|3|4 formaat 11011001 en 00101000. Vermenigvuldig deze getallen volgens het schema uit de cursus en geef het binaire resultaat in 1|3|4 formaat (alle formaten analoog aan de IEEE 754 standaard).

Vraag 2 (4 punten)

Gegeven de volgende C-functie

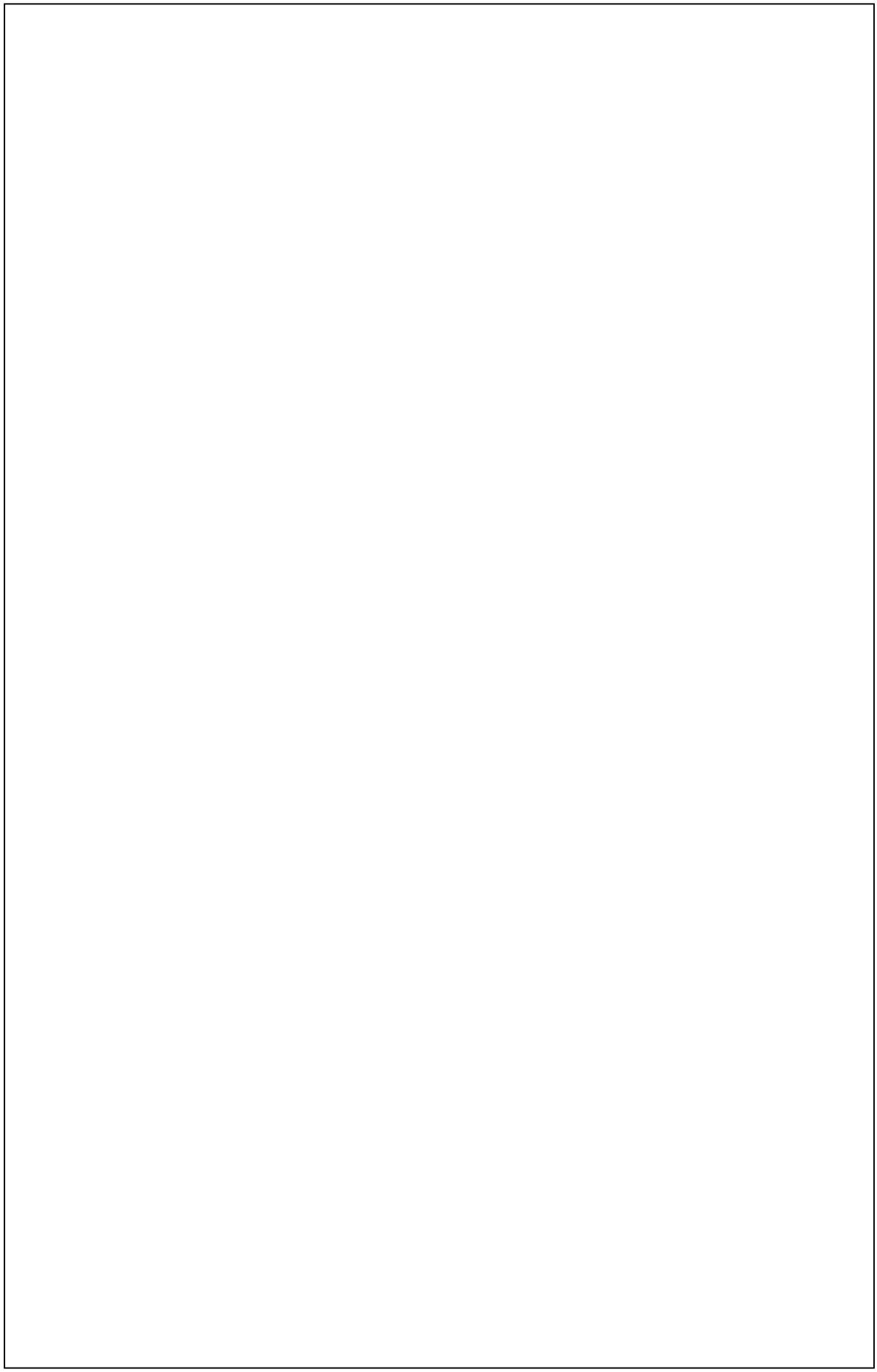
```
int zevenvoud[19] = {1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0};

int testzevenvoud(int n)
{
  if (n < 0)
    return testzevenvoud(-n);
  else if (n <= 19)
    return zevenvoud[n];
  else
    return testzevenvoud(n/10 - 2 * (n%10));
}
```

Met de volgende assemblercode

```
testzevenvoud:
      pushl %ebp
      movl  %esp, %ebp
      movl  8(%ebp), %ecx
      pushl %ebx
      movl  0x66666667, %ebx
.L15:
      testl %ecx, %ecx
      js    .L17
.L4:
      cmpl  $19, %ecx
      jle   .L9
      movl  %ecx, %eax
      imull %ebx
      sarl  $2, %edx
      leal  (%edx,%edx,4), %eax
      addl  %eax, %eax
      subl  %eax, %ecx
      leal  (%ecx,%ecx), %eax
      movl  %edx, %ecx
      subl  %eax, %ecx
      testl %ecx, %ecx
      jns   .L4
.L17:
      negl  %ecx
      jmp   .L15
.L9:
      movl  zevenvoud(,%ecx,4), %eax
      popl  %ebx
      popl  %ebp
      ret
```

Teken de controleverloopgraaf van deze functie



Waar zijn alle recursieve oproepen heen?

Wat is de betekenis van 0x66666667

Vraag 3 (4 punten)

Gegeven het volgende stukje C-code

```
#include <stdio.h>
main()
{
  int i, s = 0;
  int a[5];

  for (i=0; i<5; i++)
   s += a[i];
}
```

Met de volgende assemblercode

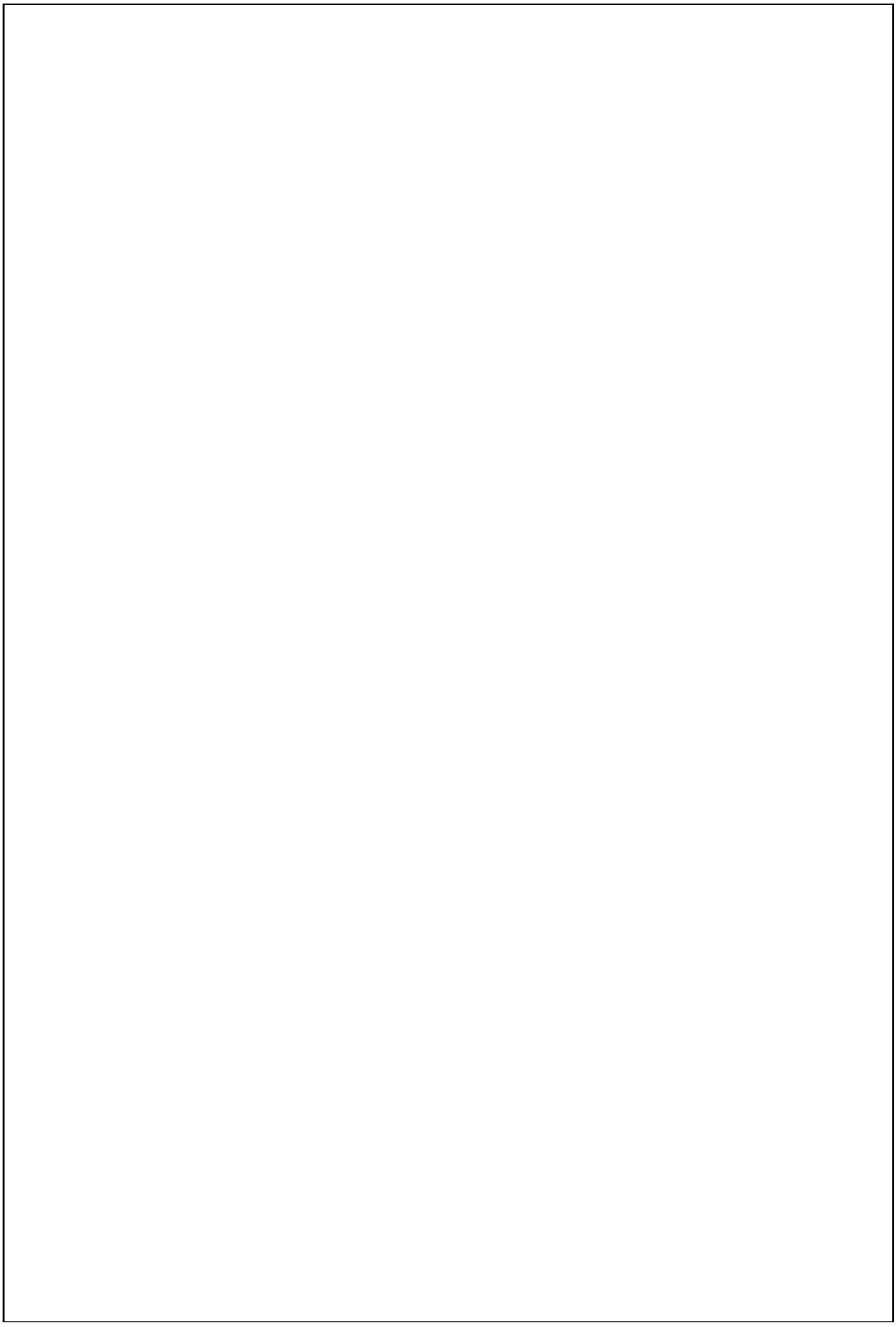
```
main:
        movl    %esp, %ebp                      (a)
        xorl    %edx, %edx                      (b)
        movl    $1, %eax                        (c)
        leal    -24(%ebp), %ecx                 (d)

.L2:
        addl    -4(%ecx,%eax,4), %edx           (e)
        addl    $1, %eax                        (f)
        cmpl    $5, %eax                        (g)
        jne     .L2                             (h)
```

Teken het verloop van de uitvoering van deze code naar het model van de superscalaire out-of-order pijplijn uit de cursus. Je mag ervan uitgaan dat de sprongen altijd correct voorspeld worden, dat er noch datacachemissers noch instructiecachemissers optreden. Een load-store instructie vergt 2 cycli in de uitvoeringsfase, de andere instructies worden allemaal in 1 cyclus uitgevoerd. De commitbandbreedte is 4 instructies.

In het instructievenster worden instructies die in uitvoering zijn onderstreept (ipv de groene kleur op de transparant), en een afgewerkte instructie die het instructievenster nog niet verlaten heeft wordt diagonaal doorstreept (ipv de rode kleur op de transparant).

Teken de volledige dataverloopgraaf van dit programma.



Vul het schema van de volgende pagina aan.

cycli	frontend								instructievenster										A	A	LS
C1	b	a																			
C2	d	c																			
C3	f1	e1																			
C4	h1	g1																			
C5	f2	e2																			
C6	h2	g2																			
C7	f3	e3																			
C8	h3	g3																			
C9	f4	e4																			
C10	h4	g4																			
C11	f5	e5																			
C12	h5	g5																			
C13																					
C14																					
C15																					
C16																					
C17																					
C18c																					
C19																					
C20																					

Vraag 4 (4 punten)

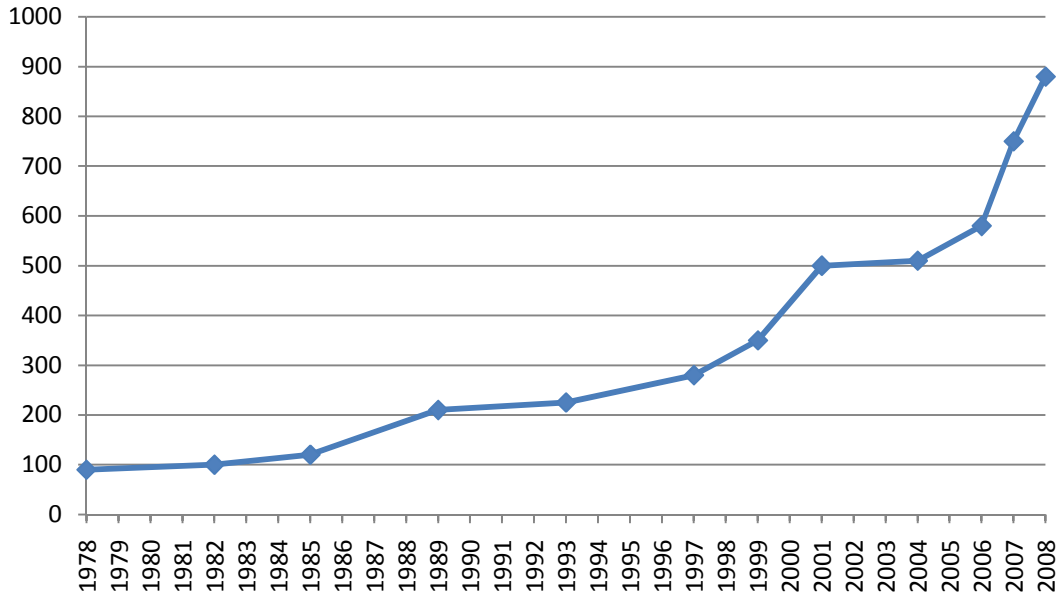
Gegeven een data cache van 4 lijnen van 8 bytes lang (write-back, write-allocate). Een programma genereert een lijst van leesopdrachten (L) of schrijfopdrachten (S) van 1 byte op de volgende (decimale) adressen (kolom *Adres*).

- Vul in de kolom Blok het bloknummer van het adres in (0,1,2,3,...).
- Vul voor Lijn de identificatie van de lijn in (b.v. a,b,c,d)
- Vul in de kolommen Inhoud de inhoud van de cache in (uitgedrukt in bloknummers) vlak na de cachetoegang (bij associativiteit steeds LRU als vervangingsstrategie gebruiken). Als de inhoud van een blok veranderd werd, schrijft u een sterretje na het bloknummer.
- Vul in de kolommen h/m in of het over een hit of een miss gaat. Indien een miss gepaard gaat met het terugschrijven van een blok, noteert u m*.
- Vul in de kolom Set de identificatie van de gebruikte set in (b.v. A of B)
- Geef voor de verschillende missers aan over welk type misser het gaat: m1 = koude misser, m2 capaciteitsmisser en m3 conflictmisser.

Opdrachten		**Direct mapped cache**			**Volledig associatief**		**2-Wegs set associatief**		
Adres	**Blok**	**Lijn**	**Inhoud**	**h/m**	**Inhoud**	**h/m**	**Set**	**Inhoud**	**h/m**
-			- | - | - | -		- - - -			- - | - -	
L 16			| | |					|	
L 2			| | |					|	
S 18			| | |					|	
S 19			| | |					|	
L 26			| | |					|	
L 32			| | |					|	
S 4			| | |					|	
S 16			| | |					|	
L 18			| | |					|	
S 4			| | |					|	
L 6			| | |					|	
S 16			| | |					|	

Vraag 5 (4 punten)

De volgende grafiek geeft de evolutie van het aantal x86 instructies weer.



Vraag: welke trend uit de HiPEAC visie wordt hierdoor geïllustreerd?

Wat is de definitie van PUE en wat zijn typische waarden voor PUE?

Op welke manier bepaalt Gordon Moore het aantal componenten dat kan geïntegreerd worden op een chip?

Wat wordt bedoeld met een vertikale of een vertikaal-geïntegreerde industrie?

cycli	frontend								instructievenster										A	A	LS
C1	b	a																			
C2	d	c																			
C3	f1	e1																			
C4	h1	g1																			
C5	f2	e2																			
C6	h2	g2																			
C7	f3	e3																			
C8	h3	g3																			
C9	f4	e4																			
C10	h4	g4																			
C11	f5	e5																			
C12	h5	g5																			
C13																					
C14																					
C15																					
C16																					
C17																					
C18c																					
C19																					
C20																					