Examen computerarchitectuur

Woensdag 21 juni 2006, 8u30

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 of rekenmachine gebruiken bij het oplossen van de vragen.

Veel succes!

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

Vraag 1 (4 punten)

Gegeven het volgende stukje assemblercode voor de IA32. Geef het resultaat van de berekening weer in de rechterkolom.

Instructie	**Resultaat (hex)**
mov eax,105	
add eax, 358h	
and eax,38fh	
rol eax,5	
or eax,100	
neg eax	
xor eax,5555h	
mul eax,5	

Vraag 2 (4 punten)

Gegeven de volgende gegevensstructuur in C.

```
struct {
    int a;
    int b[63,63];
} s;
```

en de volgende for-lus:

```
for (i=0; i<63; i++)
  s.a += s.b[i,i];
```

Hierbij is gekend dat het adres van de structuur s 100 is, en dat de waarde van i zich tijdens de uitvoering van de lus in het register ebx bevindt. Registers eax en ecx bevatten tijdens de uitvoering van de lus geen levende waarden. De grootte van een int is 32 bits.

Schrijf de adresexpressie neer die hoort bij de instructie uit de lus, en vereenvoudig deze zover mogelijk.

Schrijf de assemblerinstructie(s) voor de IA32 neer die deze adresexpressie implementeren.

Vraag 3 (4 punten)

Beschouw het volgende C-programma

```
int g;
int a[10] = {1,2,3,4,5,6,7,8,9,10};

int sum(int arr[], int s)
{
  if (s == 0)
    return 0;
  else
    return sum(arr, s-1) + arr[s-1];
}

main()
{
  g = sum(a,10);
}
```

Met als code die door de gcc-compiler gegenereerd wordt:

```
sum:
        pushl   %ebp
        movl    %esp, %ebp
        pushl   %esi
        pushl   %ebx
        movl    12(%ebp), %ebx
        xorl    %eax, %eax
        testl   %ebx, %ebx
        movl    8(%ebp), %esi
        jne     .L5
.L1:
        leal    -8(%ebp), %esp
        popl    %ebx
        popl    %esi
        leave
        ret
.L5:
        leal    -1(%ebx), %eax
        pushl   %eax
        pushl   %esi
        call    sum
        addl    -4(%esi,%ebx,4), %eax
        jmp     .L1
```

```
a:      .long   1
        .long   2
        .long   3
        .long   4
        .long   5
        .long   6
        .long   7
        .long   8
        .long   9
        .long   10
main:
        pushl   %ebp
        movl    %esp, %ebp
        pushl   $10
        pushl   $a
        call    sum
        popl    %edx
        popl    %ecx
        movl    %eax, g
        leave
        ret
```

Teken de controleverloopgraaf van dit programma (1 punt)

Verklaar de volgende instructie (1 punt)

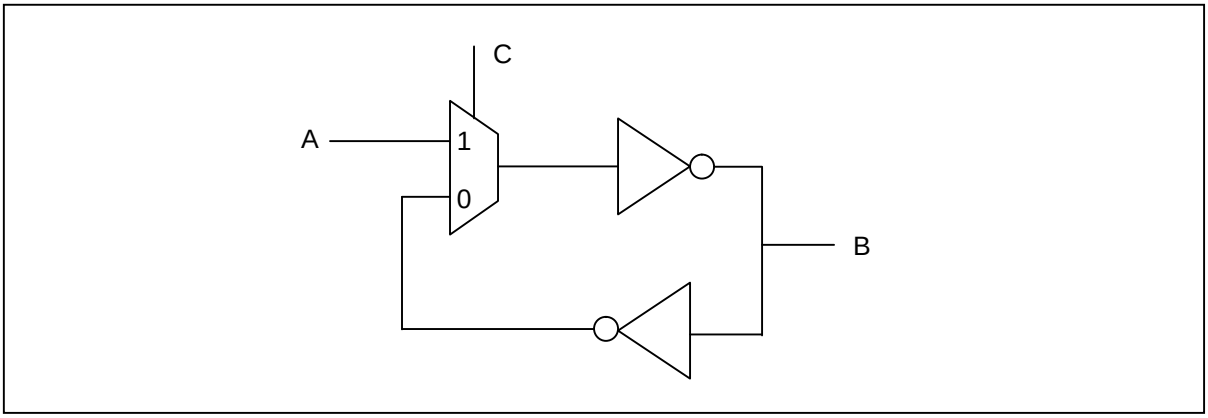
```
addl -4(%esi,%ebx,4), %eax
```

Teken de stapel op het ogenblik dat de instructie `pushl %ebx` voor de tweede keer uitgevoerd werd (1 punt)

Hoeveel instructies worden er in het totaal door dit programma uitgevoerd (1 punt)?

Vraag 4 (4 punten)

Gegeven de volgende schakeling



Geef het gedrag van deze schakeling weer aan de hand van een waarheidstabel (2 punten).

Beschrijf het gedrag van deze schakeling in woorden. Komt dit gedrag overeen met een bekende bouwsteen? (1 punt)

Wat is de totale vertraging die geassocieerd kan worden met deze schakeling? Ga ervan uit dat de drie componenten uit de schakeling individueel één poortvertraging veroorzaken (1 punt).

Vraag 5 (4 punten)

Gegeven het volgende escape-programma:

```
0000: 44010100 |            | ADDI R0, 0x0100, R1    | 5 |   |
0004: 4403000A |            | ADDI R0, 0x0002, R3    | 6 |   |
0008: 20421000 |            | SUB  R2, R2, R2        |   |   |
000C: 0C250000 | target     | LDW  R5, 0x0000(R1)    |   |   |
0010: 1C451000 |            | ADD  R2, R5, R2        |   |   |
0014: 44210004 |            | ADDI R1, 0x0004, R1    |   |   |
0018: 48630001 |            | SUBI R3, 0x0001, R3    |   |   |
001C: 7803FFEC |            | BRGT R3, target        |   |   |
0020: 18020100 |            | STW  R2, 0x0100(R0)    |   |   |
```

Ga ervan uit dat de toegangstijd naar het geheugen 2 cycli bedraagt, dat forwarding aan staat, en dat de delay slots af staan.

Schrijf in de vierde en vijfde kolom in de hoeveelste cyclus deze instructie zich in de WB trap bevindt (naar het voorbeeld van de eerste twee instructies).

Herschrijf het programma zodat het zo snel mogelijk uitgevoerd wordt. Hierbij mag U instructies willekeurig verschuiven, maar niet weglaten. U mag ook de parameters van de architectuur wijzigen (delay slots, forwarding).

In de hoeveelste cyclus komt de laatste instructie STW nu in de WB-trap?