Student: **Collegekaartnummer:**

Tentamen Computersystemen

baiCOSY06 2e jaar bachelor AI, 2e semester 27 september 2012 13u-15u

IWO 4.04C (rood), Academisch Medisch Centrum, Meidreef 29, Amsterdam ZuidOost

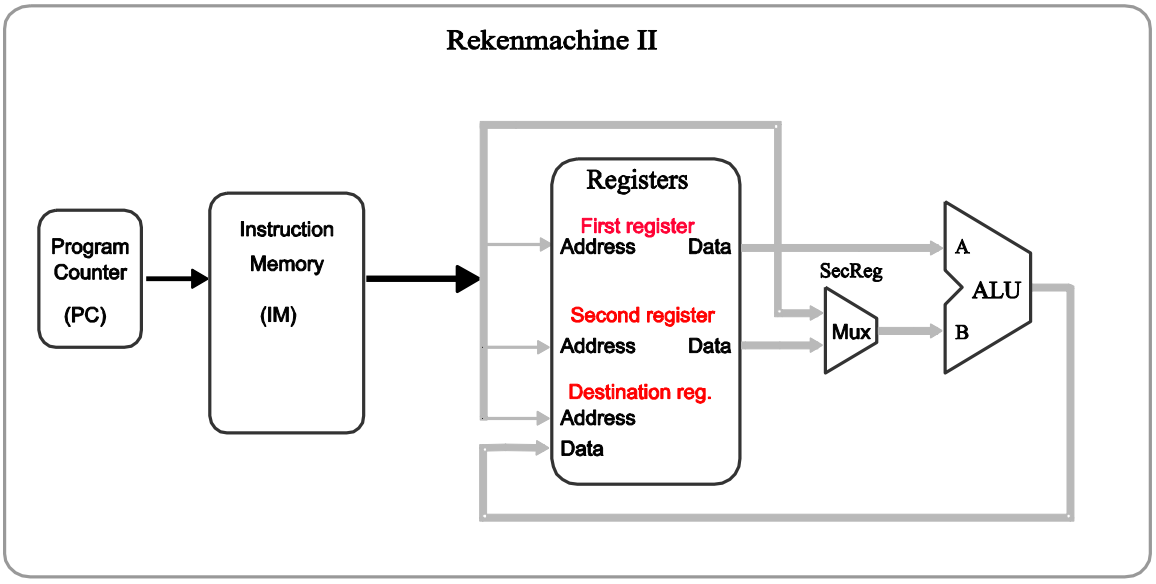
Het is niet toegestaan communicatieapparatuur zoals tablets en telefoons te gebruiken: zet de mobiele telefoon uit!

Gebruik van een rekenmachine en de boeken behorende bij dit vak (Computer Systems, en Van 0 en 1 tot processor) is toegestaan. Succes!

vraag 1

Vraag 1 Rekenschakelingen
Vraag 1a: Wat is een Full Adder?
Vraag 1b: Geef de waarheidstabel van een Full Adder weer.

Vraag 2 Rekenmachine II
Vraag 2a: Wat is de opcode van een instructie?
Vraag 2b: Arceer het datapad van een ADDI-instructie in figuur 1 Geef ook aan welke adresingang(en) worden gebruikt bij deze instructie.



Figuur 1: Rekenmachine II

Vraag 2c: Welke component(en) hebben een kloksignaal nodig om de Rekenmachinemachine II goed te laten werken?

Student: **Collegekaartnummer:**

Vraag 3 Harvard machine *Zie figuur 2*.

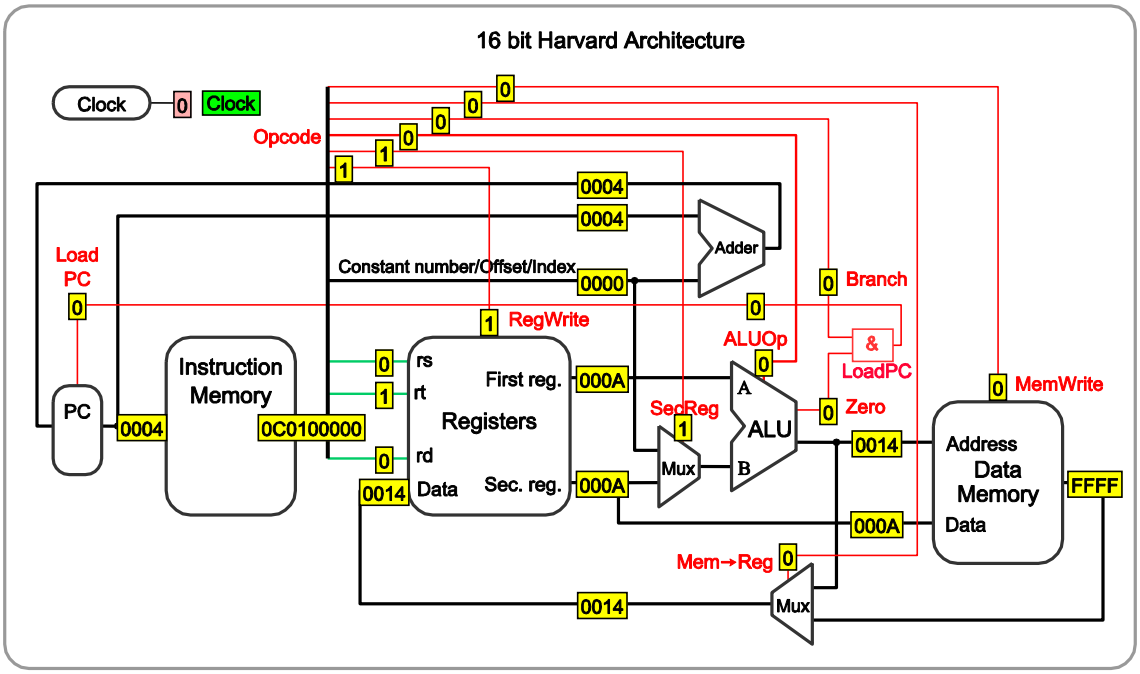
Vraag 3a: Uit welke fases bestaat het executeren van een “Store Word”-instructie?

Vraag 3b: Wat is de functie van de component registers?

Vraag 3c: Noem alle instructies waarbij tijdens het executeren ervan het SecReg-bit 0 is? Licht je antwoord toe.

Vraag 3d: Welke instructie wordt in figuur 2 uitgevoerd? Licht je antwoord toe. Geef de syntax van deze instructie weer.

Vraag 3e: Wat is de functie van de component “Adder”? Licht je antwoord toe.

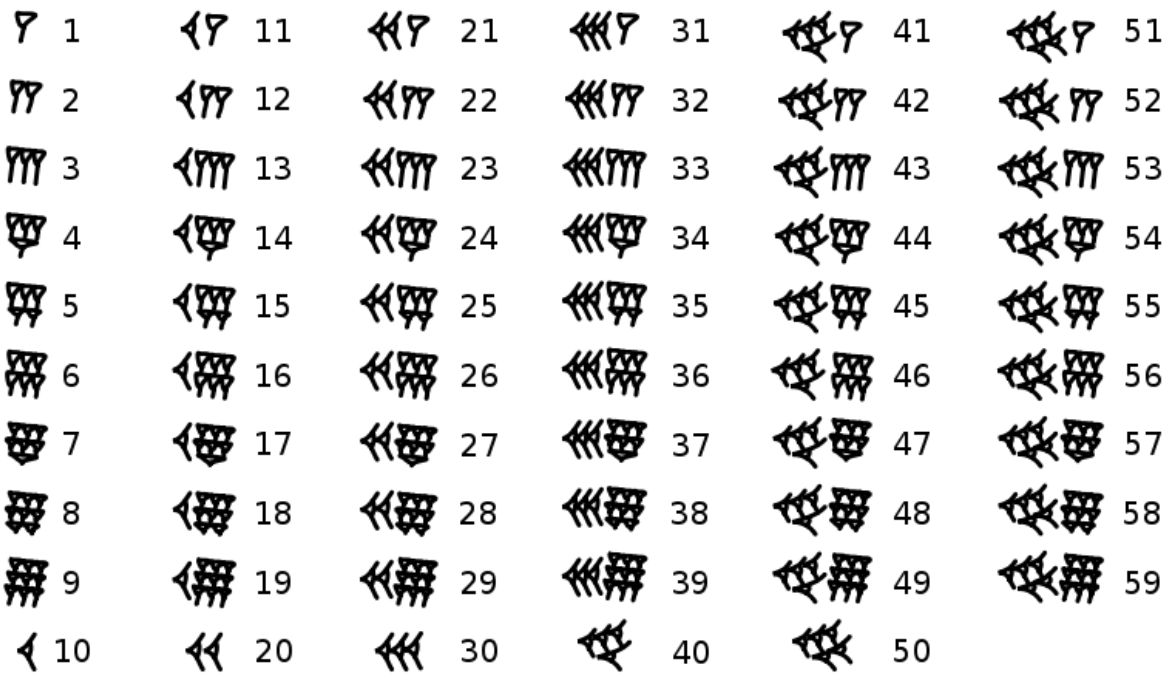


Figuur 2: Harvard machine

Student: **Collegekaartnummer:**

vraag 4

De Babeloniërs gebruikten een zestigtallig stelsel door twee symbolen te combineren;
voor de eenheden en voor de tientallen:



De Babeloniërs maakten grotere getallen door ze naast elkaar te zetten. Het begrip nul was in 3000 v.Chr. nog niet uitgevonden, daarvoor in de plaats gebruikten ze een spatie. Een voorbeeld is de volgende berekening, waar men de ruimte tussen de 6 en de 9 duidelijk groter is dan de ruimte tussen de 2 en de 27:

$$147^2 = 21609$$
$$(2x60^1 + 27x60^0)^2 = (6x60^2 + 9x60^0)$$

Om verwarring te voorkomen, plaatst men tegenwoordig komma's of dubbele punten tussen de Babelonische getallen.

a. Geef de formule **b2U$_w$** *("Babelonian to unsigned", length* ***w***) voor het 60-talig stelsel van de Babeloniers.

Student: **Collegekaartnummer:**

De Babeloniers kende het gegrip *floating point* ook al, hoewel ze ook daar net als voor de nul geen notatie voor hadden. Tegenwoordig zetten we een punt-komma op de juiste plaats in Babelonische getallen. Bijvoorbeeld; de Babeloniers schatten π af op 25/8 (3.125)[1], dus in hun notatie (3;7:30) = $(3x60^0 + 7x60^{-1} + 30x60^{-2})$:

b. Breid de functie $\mathbf{b2U_w}$ uit met negatieve indices tot de formule $\mathbf{b2F_w}$ *("Babelonian to unsigned float", length* ***w***) voor het 60-talig stelsel van de Babeloniers.

c. Geef aan of de Babelonische schatting van π (25/8) in een Bineair *signed floating point* formaat van een enkele Byte (8 bits) past. Zo ja, geef een voorbeeld van zo'n 8bits *floating point* formaat representatie van 25/8. Zo nee, toon aan waarom dit getal niet exact gerepresenteerd kan worden.

De Babeloniers gebruikte de reciproke van 7 nauwelijks. Archimedes vond met behulp van de meerzijdige polygonen (96 zijden) de eerste convergent van π: 22/7.

d. Geef aan of de schatting van Archimedes (22/7) in een Bineair *signed floating point* formaat van een enkele Byte (8 bits) past. Zo ja, geef een voorbeeld van zo'n 8bits *floating point* formaat representatie van 22/7. Zo nee, toon aan waarom dit getal niet exact gerepresenteerd kan worden.

De Chinesen gebruikten een vergelijkbare methode als Archimedes om π af te schatten. In 480 na Christus vond Zu Chongzhi (**祖沖之**) de convergent 355/113 met behulp van een polygoon met 12288 zijden. Deze schatting staat bekend onder de naam **Milü** (**密率**). Deze breuk is in de toenmalige Chinese notatie geschreven als:

Pas in de 15e eeuw kwam Jamshid al-Kashi (غیاث الدین جمشید کاشانی) met een nauwkeuriger schatting (m.b.v. een polynoom met $3x2^{28}$ zijden). In de 16e eeuw was het wereldrecord trouwens in Nederlandse handen (Ludolph van Ceulen), waarbij π tot 35 decimale cijfers achter de komma bekend was.

e. Het is jouw taak om **Milü** zo nauwkeurig mogelijk in een Bineair *signed floating point* formaat van een dubbele Byte (16 bits) te passen. Je hebt hierbij de keuze hoeveel bits je gebruikt voor de velden *exp* en *frac*: *k=5* en *f=10* of *k=4* en *f=11* of *k=3* en *f=12*. Wat is de stapgrote van het *normalized floating point* formaat rond het getal 355/113 voor deze drie keuzes? Laat met een berekening zien hoe je op deze stapgrote komt.

f. Wat is de *range* van een 16 bits *signed floating point* formaat voor de drie keuzes van *exp* en *frac*: *k=5* en *f=10* of *k=4* en *f=11* of *k=3* en *f=1?*. Oftewel, wat is de binaire en decimale waarde van het grootste *normalized floating point* getal voor deze drie keuzes?

g. Gebruik makend van de inverse van de formule die je bij vraag **b** hebt ontwikkeld, wat is de Babelonische representatie van **Milü**? Gaarne een antwoord dat nauwkeurig is drie cijfers achter de Babelonische komma.

[1] http://pidayinternational.org/pi_history/history_of_pi_babylon

Student: **Collegekaartnummer:**

vraag 5

De volgende code wordt uitgevoerd op een machine waar het type `int` een 32-bit *two's component* representatie heeft. Variabelen van het type `float` en `double` worden op deze machine respectivelijk met het 32-bit en 64-bit *IEEE format* gerepresenteerd. U kunt de bijzonderheid van de IA32 architectuur (die intern een 80 bit representatie gebruikt) voor deze opgave even vergeten.

In de code worden eerst willekeurige *integer* waarden gegeneerd, die vervolgens naar het type `double` worden geconverteerd:

```
1      /* Create some random values */
2      int x = random();
3      int y = random();
4      int z = random();
5      /* convert to double */
6      double dx = (double )x;
7      double dy = (double )y;
8      double dz = (double )z;
```

Geef voor de volgende stukjes C-code aan of de test **in alle gevallen** waar is, of niet. Indien naar u mening **waar**, dient u dat te ondersteunen met een afleiding uit de onderliggende wiskundige eigenschappen. Indien naar u mening **niet** (in alle gevallen) **waar**, dient u dit aan te tonen door een voorbeeld te geven wanneer de test een 0 oplevert:

```
A      (double )(float )x == dx
B      dx + dy == (double )(x+y)
C      dx + dy + dz == dz + dy + dx
D      dx * dy * dz == dz * dy * dx
E      dx / dx == dy / dy
```

Student: **Collegekaartnummer:**

vraag 6

U heeft een re-engineering taak gekregen: u dient de C-code te reconstrueren die hoort bij de volgende definities, en de IA32 *assembly* code die ontstaan is bij het compileren van de oorspronkelijk C code.

```
struct s1 {
   char a[3];
   union u1 b;
   int c;
};
```

```
struct s2 {
   struct s1 *d;
   char e;
   int f[4];
   struct s2 *g;
};
```

```
union u1 {
   struct s1 *h;
   struct s2 *i;
   char j;
};
```

Zoals u weet is de grootte van een `char` 1 byte, en een `int` 4 bytes Het is misschien handig om configuratie van de drie data-structuren hieronder te tekenen:

Student: **Collegekaartnummer:**

Vul nu voor elk van de vier C-functies aan de rechterkant de missende code in, aan de hand van de *assembly-code* aan de linkerkant. De instructie `movsbl` kopieert een enkele byte naar een 32 bits adres, en vult de andere 3 bytes op met de *sign*-bit van de gekopieerde byte:

A.
```
proc1:
    pushl %ebp
    movl %esp,%ebp
    movl 8(%ebp),%eax
    movl 12(%eax),%eax
    movl %ebp,%esp
    popl %ebp
    ret
```
```
int proc1(struct s2 *x)
{
  return x->____________________ ;
}
```

B.
```
proc2:
    pushl %ebp
    movl %esp,%ebp
    movl 8(%ebp),%eax
    movl 4(%eax),%eax
    movl 20(%eax),%eax
    movl %ebp,%esp
    popl %ebp
    ret
```
```
int proc2(struct s1 *x)
{
  return x->____________________ ;
}
```

C.
```
proc3:
    pushl %ebp
    movl %esp,%ebp
    movl 8(%ebp),%eax
    movl (%eax),%eax
    movsbl 4(%eax),%eax
    movl %ebp,%esp
    popl %ebp
    ret
```
```
char proc3(union u1 *x)
{
  return x->____________________ ;
}
```

D.
```
proc4:
    pushl %ebp
    movl %esp,%ebp
    movl 8(%ebp),%eax
    movl (%eax),%eax
    movl 24(%eax),%eax
    movl (%eax),%eax
    movsbl 1(%eax),%eax
    movl %ebp,%esp
    popl %ebp
    ret
```
```
char proc4(union u1 *x)
{
  return x->____________________ ;
}
```

Succes!