

Student:

Collegekaartnummer:

## Tentamen Computersystemen voor AI programmeurs

*baiCSA13 2e jaar bachelor AI, 1e semester 27 oktober 2005*

### Vraag 1 (40 ptn)

Gegeven de volgende assembly code:

```
00000000 <_sum_loop>:
 0: 55          push   %ebp
 1: 89 e5      mov    %esp,%ebp
 3: 83 ec 08   sub   $0x8,%esp
 6: c7 45 f8 16 00 00 00  movl  $0x16,0xffffffff8(%ebp)
 d: c7 45 fc 09 00 00 00  movl  $0x9,0xffffffffc(%ebp)
14: 8b 45 fc   mov   0xffffffffc(%ebp),%eax
17: 3b 45 10   cmp   0x10(%ebp),%eax
1a: 7c 02     jl    1e <_sum_loop+0x1e>
1c: eb 1c     jmp   3a <_sum_loop+0x3a>
1e: 8b 4d 08   mov   0x8(%ebp),%ecx
21: 8b 55 0c   mov   0xc(%ebp),%edx
24: 89 d0     mov   %edx,%eax
26: 01 c0     add   %eax,%eax
28: 01 d0     add   %edx,%eax
2a: 89 c2     mov   %eax,%edx
2c: 03 11     add   (%ecx),%edx
2e: 8d 45 f8   lea  0xffffffff8(%ebp),%eax
31: 01 10     add   %edx,(%eax)
33: 8d 45 fc   lea  0xffffffffc(%ebp),%eax
36: ff 00     incl (%eax)
38: eb da     jmp   14 <_sum_loop+0x14>
3a: 8b 45 f8   mov   0xffffffff8(%ebp),%eax
3d: c9       leave
3e: c3       ret
```

De assembly code werd gegenereerd door de volgende C-code te compileren met de '-c' optie van gcc:

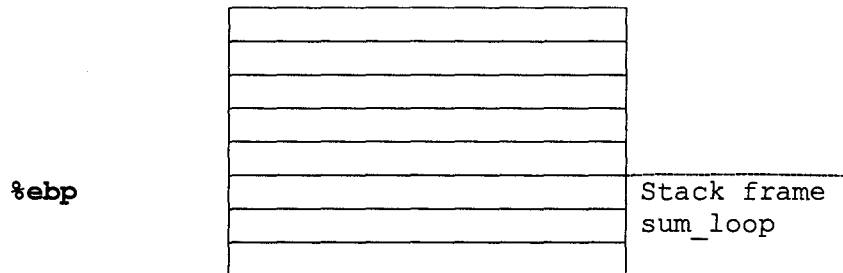
```
int sum_loop( int *x, int y, int n ) {
    int i;
    int result = .....;

    for( .....; .....; i++ ) {
        result +=.....;
    }
    return result;
}
```

Student:

Collegekaartnummer:

1a) (10 ptn) Vul de stack in zoals deze eruit ziet bij aanvang van de instructie op geheugen adres 14.



1b) (5 ptn) Na de compare instructie op geheugenadres 17 (`cmp s2, s1` staat voor `s1-s2`) volgt een jump less instructie. Deze instructie maakt gebruik van de sign flag SF en de overflow flag OF. In welk van de gevallen wordt jump less instructie uitgevoerd?

SF	OF	jl
0	0	
0	1	
1	0	
1	1	

1c) (5 ptn) Bespreek de byte codering `eb 1c` van de assembly-instructie `jmp 3a <_sum_loop+0x3a>`

1d) (5 ptn) Compileren, 'gcc -c loop.c', levert de file loop.o op.

Dit is:

- A Een relocatable object file gegenereerd door de C preprocessor
- B Een shared object file gegenereerd door de assembler (as).
- C Een relocatable object file gegenereerd door de assembler (as) uit het tijdelijke bestand loop.s
- D Een executable object file gegenereerd door de linker.

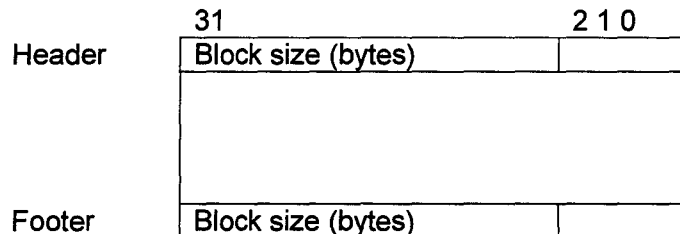
1e) (15 ptn) Vul de ontbrekende C-code in ( `lea` staat voor load effective address)

Student:

Collegekaartnummer:

Vraag 2 (35 ptn)

Gegeven een memory allocator die gebruik maakt van een implicit free list.



Elk geheugenblok is een veelvoud van 2 geheugenwoorden (8 bytes!). Dus de 3 minst significante bits van de header en footer van een geheugenblok zijn vrij en worden als volgt gebruikt:

- bit 0: geeft aan of het geheugenblok vrij of bezet is; 1 voor bezet, 0 voor vrij.
- bit 1: geeft aan of het vorige geheugenblok vrij of bezet is; 1 voor bezet en 0 voor vrij.
- bit 3: ongebruikt

2a) (5 ptn) Gegeven een functie die een pointer naar de header van een geheugenblok teruggeeft. Het argument (void \*p) is een pointer naar de payload van het geheugenblok zoals dat door malloc wordt teruggegeven. Welk codefragment, A, B of C, hoort op de ontbrekende plaats?

```
void * header(void* p)
{
    void *ptr;
    .....;
    return ptr;
}
```

- A. ptr=p-1
- B. ptr=(void \*)((int \*)p-1)
- C. ptr=(void \*)((int \*)p-4)

2b) (5 ptn) Gegeven een functie die de size van een geheugenblok teruggeeft. Welk codefragment, A, B of C, hoort op de ontbrekende plaats?

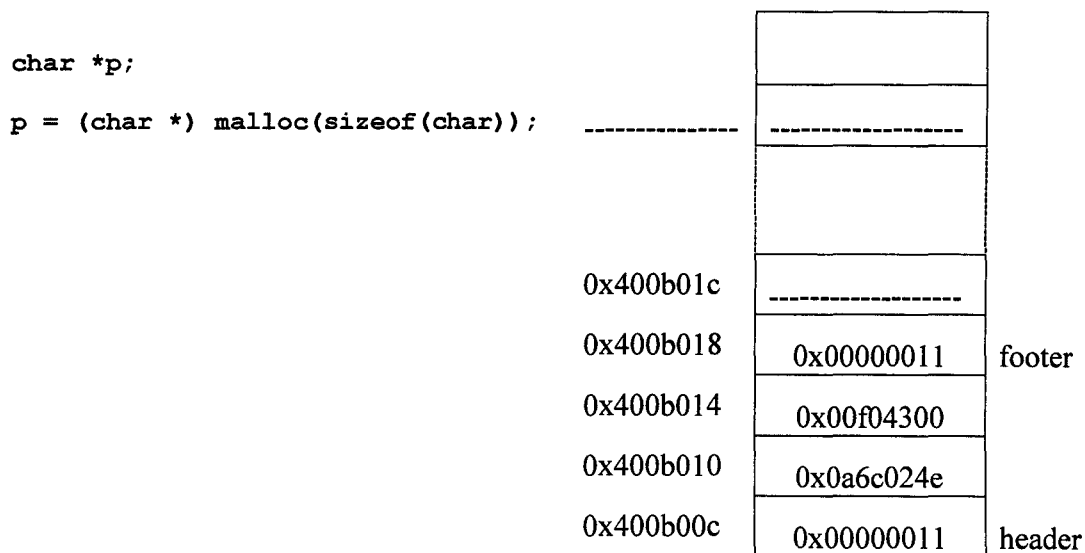
```
int size(void *hp)
{
    int result;
    .....;
    return result;
}
```

- A. result=(\*hp) & (~7)
- B. result=((\*(char \*)hp) & (~5)) << 2
- C. result=\*(int \*)hp & (~7)

Student:

Collegekaartnummer:

2c) (15 ptn) Gegeven de onderstaande heap, met op adres 0x400b018 de footer van een bezet geheugenblok. Het codefragment bevat een geheugenallocatie voor een enkele byte. Geef aan hoe de heap er uit ziet indien de allocator een geheugenblok alloceert op adres 0x400b01c. Vul de header en footer in van het nieuwe blok en het adres van de footer (de heap groeit van onder naar boven). Lees goed de gegevens aan het begin.



2d) (10 ptn) Iemand komt met de volgende wrapper functie voor het vrijmaken van geheugen:

```
void free_memory( void *p )  
{  
    pid_t pid;  
  
    pid = fork();  
    if( pid == 0 ) {  
        free( p );  
        exit(0);  
    }  
  
    coalesce( p );  
    return;  
}
```

Geef meerdere redenen waarom deze implementatie onzin is.

Student:

Collegekaartnummer:

### Vraag 3 (10 ptn)

Gegeven de volgende twee bestanden:

```
/* foo.c */
#include <stdio.h>
void f( void );

int x = 15213;
int y = 15212;

int main( )
{
    f();
    printf( "x = 0x%x y = 0x%x \n"
           x, y );
    return 0;
}
```

```
/* bar.c */
double x;

void f( )
{
    x = -0.0;
}
```

Compilieren en runnen geeft de volgende output:

```
> gcc -o foo foo.c bar.c
> ./foo
x = 0x0 y = 0x80000000
```

Geef uw commentaar: