Universidad de los Andes

Ingeniería de Sistemas y Computación
ISIS1304 – Fundamentos de Infraestructura Tecnológica
Banco preguntas - Datos Compuestos

Capacidades evaluadas:

- *Calcular tamaños (escalar, vector, estructura)*
- *Calcular Desplazamientos (vector, estructura) - relativos al principio del dato*
- *Calcular direccionamiento absoluto de la memoria*
- *Hacer conversiones (hexa, unidades)*

Recordatorio:
Tamaño de un vector = número de elementos * tamaño de cada elemento
Tamaño de una estructura = suma de los tamaños de los campos
Desplazamiento subíndice i en vector = i * tamaño de cada elemento
Desplazamiento i-ésimo campo de estructura = sumatoria de tamaños campos anteriores
Dirección de un campo i en un dato compuesto = dirección inicial dato compuesto + desplazamiento hasta i desde el comienzo del dato compuesto

---

Se tienen las siguientes declaraciones en C:

```
struct R {
  char a;
  int b[10];
  short c;
};

struct S {
  int x;
  struct R sr;
  char y[15];
  short z;
};

struct S   v[90] ;
```

1. ¿Cuánta memoria ocupa el vector (expréselo en Bytes y en Kibits)?

Tamaño de `int [10]` = 10*tamaño de `int` = 10*4 Bytes = 40 Byte

Tamaño de `struct R` = tamaño de `char` + tamaño de `int [10]` + tamaño de `short` = (1 + 40 + 2) Byte = 43 Byte

Tamaño de `struct S` = tamaño de `int` + tamaño de `struct R` + tamaño de `char [15]` + tamaño de `short` = (4 + 43 + 15 + 2) Byte = 64 Byte

Tamaño de `struct S [90]` = 90 * tamaño de `struct S` = 90 * 64 Byte = 5760 Byte = 5760*8 bits = 46080 bits

1 Kibit = 1024 bit ⇒ bit = 1 Kibit / 1024 ⇒
46080 bit = 46080 Kibit /1024 = 45 Kibit

2. Si el vector inicia en la dirección `0x2A0`, ¿en qué dirección de memoria (hexadecimal) se encuentra el elemento `v[20].sr.b[3]`?

Desplazamiento hasta `v[20]` (desde el inicio de `v`) = 20* tamaño `struct S` = 20*64 Byte = 1280 Byte

Desplazamiento hasta campo `sr` (desde el comienzo de la estructura S) = tamaño de `int` = 4 Byte

Desplazamiento hasta campo `b` (desde el comienzo de la estructura R) = tamaño de `char` = 1 Byte

Desplazamiento hasta `b[3]` (desde el comienzo de `b`) = 3*tamaño `int` = 12 Byte

Luego, desplazamiento hasta `v[20].sr.bc[3]` (desde el inicio de `v`) = (1280+4+1+12) Byte = 1297 Byte

Por ende, dirección de `v[20].sr.bc[3]` = dirección inicial de `v` + desplazamiento de `v[20].sr.bc[3]` = 0x10C + 1297 = 0x2A0 + 0x511 = 0x7B1

Nota:

```
1297 | 16
   1  81 | 16
       1  5 |
```

Luego, 1297 = 0x511

Se tienen las siguientes declaraciones en C:

```
struct S {
  int a;
  char b;
  short c[30];
  short d;
};

struct S   v[120] ;
```

1. ¿Cuánta memoria ocupa el vector (expréselo en bytes y en KiBytes)?

Tamaño de `short [30]` = 30*tamaño de `short` = 30*2 Bytes = 60 Byte

Tamaño de `struct S` = tamaño de `int` + tamaño de `char` + tamaño de `short [30]` + tamaño de `short` = (4 + 1 + 60 + 2) Byte = 67 Byte

Tamaño de `struct S [120]` = 120 * tamaño de `struct S` = 120 * 67 Byte = 8040 Byte

1 KiByte = 1024 Byte ⇒ Byte = 1 KiByte / 1024 ⇒
8040 Bytes = 8040 KiByte/1024 = 7,85 KiByte

2. Si el vector inicia en la dirección `0x10C`, ¿en qué dirección de memoria (hexadecimal) se encuentra el elemento `v[40].c[5]`?

Desplazamiento hasta `v[40]` (desde el inicio de `v`) = 40* tamaño `struct S` = 40*67 Byte = 2680 Byte

Desplazamiento hasta campo `c` (desde el comienzo de la estructura) = tamaño de `int` + tamaño de `char` = (4 + 1) Byte = 5 byte

Desplazamiento hasta `c[5]` (desde el comienzo de `c`) = 5*tamaño `short` = 5*2 Byte = 10 Byte

Luego, desplazamiento hasta `v[40].c[5]` (desde el inicio de `v`) = (2680+5+10) Byte = 2695 Byte

Por ende, dirección de `v[40].c[5]` = dirección inicial de `v` + desplazamiento de `v[40].c[5]` = 0x10C + 2695 = 0x10C + 0xA87 = 0xB93

Nota:

```
2695 | 16
   7 | 168 | 16
       8 | 10 |
```

Luego, 2695 = 0xA87

```
struct T {
  short a;
  char b[10];
  int c;
};

struct R {
  char x[10];
  struct T t[40];
  char y;
  short z;
};

struct R   s;
```

1. ¿Cuánta memoria ocupa la variable `s` (expréselo en Bytes y en Kibits)?

Tamaño de `char [10]` = 10*tamaño de `char` = 10*1 Bytes = 10 Byte

Tamaño de `struct T` = tamaño de `short` + tamaño de `char [10]` + tamaño de `int` = (2 + 10 + 4) Byte = 16 Byte

Tamaño de `struct T [40]` = 40*tamaño de `struct T` = 40*16 Bytes = 640 Byte

Tamaño de `struct R` = tamaño de `char [10]` + tamaño de `struct T [40]` + tamaño de `char` + tamaño de `short` = (10 + 640 + 1 + 2) Byte = 653 Byte

Tamaño de `S` = Tamaño de `struct R` = 653 Byte = 653*8 bits = 5224 bits

1 Kibit = 1024 bit ⇒ bit = 1 Kibit / 1024 ⇒ 5224 bit = 5224 Kibit /1024 = 5,1 Kibit

2. Si la variable `s` se encuentra en la dirección `0x1B4`, ¿en qué dirección de memoria (hexadecimal) se encuentra el elemento `s.t[5].b[3]`?

Desplazamiento hasta `t[5]` (desde el inicio de `s`) = tamaño `char [10]` + 5* tamaño `struct T` = (10 + 5*16) Byte = 90 Byte

Desplazamiento hasta campo `b[3]` (desde el comienzo de la estructura T) = tamaño de `short` + 3*tamaño `char` = (2 + 3) Byte = 5 byte

Luego, desplazamiento hasta `s.t[5].b[3]` (desde inicio de `s`) = (90+5) Byte = 95 Byte

Por ende, dirección de `s.t[5].b[3]` = dirección inicial de `s` + desplazamiento de `s.t[5].b[3]` = 0x1B4 + 95 = 0x1B4 + 0x5F = 0x213

Nota:

95 | 16
15 5

Luego, 95 = 0x5F

Se tienen las siguientes declaraciones en C:

```
struct PIXEL {
  char transparencia;
  char rgb[3];
};

struct IMAGEN {
  int ancho;
  int alto;
  struct PIXEL bitmap[100][80];
};

struct VIDEO {
  char nombre[20];
  short nFramesSec;
  struct IMAGEN frame[250];
};

struct VIDEO videoteca[300];
```

1. ¿Cuánta memoria ocupa una `IMAGEN` (expréselo en Bytes)?

Tamaño struct PIXEL = Tamaño de char + tamaño char[3] = 1 + (1 * 3) = 4 bytes

Tamaño struct IMAGEN = Tamaño de int + tamaño de int + tamaño PIXEL[100][80] =

4 + 4 + (100 * 80 * Tamaño struct PIXEL) = 4 + 4 + (100 * 80 * 4) = 32008 bytes

2. ¿Cuánta memoria ocupa la `videoteca` (expréselo en MiBytes)?

Tamaño struct VIDEO = Tamaño char[20] + Tamaño de short + tamaño de IMAGEN[250] = (1 * 20) + 2 + (32008 * 250) = 8002022 bytes

Tamaño videoteca[300] = 300*Tamaño VIDEO = 300 * 8002022 = 240060600 bytes

Pasar a MyBytes

240060600 bytes / 2^{20} = 240060600 bytes / 1024^2 = 2289,39 MiBytes

3. Dado que el vector comienza en la dirección 0xC80B4, ¿en qué dirección de memoria (expresada en hexadecimal) empieza el elemento `videoteca[15].frame[8].bitmap[25][5]`?

Pasar posición inicial de hexa a decimal

$(C80B4)_{16} = (819380)_{10}$

Desplazamiento hasta `videoteca[15]` (desde el inicio de `videoteca`) = 15* tamaño `struct VIDEO` = 15*8002022 Byte = 120030330 Byte

Desplazamiento hasta campo `frame` (desde el comienzo de la estructura `VIDEO`) = tamaño `char[20]` +tamaño `short` = (1*20+2) Byte = 22 Byte

Desplazamiento hasta `frame[8]` (desde el comienzo de `frame`) = 8 * tamaño `IMAGEN` = 8 *32008 Byte = 256064 Byte

Desplazamiento hasta campo `bitmap` (desde el comienzo de la estructura `IMAGEN`) = tamaño `int` +tamaño `int` = (4+4) Byte = 8 Byte

Desplazamiento hasta `bitmap[25]` (desde el comienzo de `bitmap`) = 25 * tamaño `PIXEL[80]` = 25 *80*4 Byte = 8000 Byte

Desplazamiento hasta `bitmap[25][5]` (desde el comienzo de `bitmap[25]`) = 5 * tamaño `PIXEL` = 5 *4 Byte = 20 Byte

Desplazamiento total =(120030330 + 22 + 256064 + 8 + 8000 + 20) Byte = 120294444 byte

Dirección = dirección inicial + desplazamiento total = (819380 + 120294444) Bytes = 121113824 bytes

Se pasa el resultado decimal a hexa:

$(121113824)_{10} = (7380CE0)_{16}$

Se tienen las siguientes declaraciones en C:

```
struct IDENTIFICACION {
  char genero;
  int nombre[20];
  int cedula;
};

struct EMPLEADO {
  int salario;
  struct IDENTIFICACION id;
  char horasMes[12];
```

```
  short retencion;
};

struct EMPLEADO   nomina[150] ;
```

1. ¿Cuánta memoria ocupa el vector (expréselo en Bytes y en Kibits)?

Tamaño struct IDENTIFICACION:

Tamaño de char + Tamaño de arreglo de int + Tamaño de int

1 + (4*20) + 4 = 85 bytes

Tamaño struct EMPLEADO:

Tamaño de int + Tamaño de struct + tamaño de arreglo de char + tamaño de short

4 + 85 + (12*1) + 2 = 103 bytes

Tamaño arreglo nomina

103 * 150 = 15450 bytes

Pasar a Kibits:

(15450 * 8) / 1024 = 120,7 Kibits

2. Si el vector inicia en la dirección `0x10B4`, ¿en qué dirección de memoria (hexadecimal) se encuentra el elemento `nomina[30].id.nombre[10]`?

Pasar posición inicial de hexa a decimal

$(10B4)_{16} = 1*16^3 + 11*16^1 + 4*16^0 = 4276$

Se suman 30 veces el tamaño de empleado, Se le suma el int de salario antes de llegar a id, Se suma el char del género y por último se suman 10 veces el int de nombre.

4276 + (103*30) + 4 + 1 + (4*10) = 7411

Se pasa el resultado decimal a hexa

$(7411)_{10} = (1CF3)_{16}$

Nota:

```
7411 | 16
   3   463 | 16
         15   28 | 16
              12   1
```

Luego, 7411 = 0x1CF3

```
struct MUESTREO {
  char nBits;
  int valor;
};

struct CANAL {
  int muestreosSeg;
  int nMuestreos;
  struct MUESTREO muestra[1000000];
```

```
};
struct PISTA {
  char nombre[20];
  short nCanales;
  struct CANAL canales[5];
};
struct PISTA audioteca[50];
```

1. ¿Cuánta memoria ocupa `CANAL` (expréselo en Bytes)?

Tamaño struct MUESTREO = tamaño char + tamaño int = (1 + 4) byte = 5 bytes

Tamaño struct CANAL = Tamaño int + tamaño int + tamaño MUESTREO[1000000] = 4 + 4 + (1000000 * Tamaño struct MUESTREO) = 4 + 4 + (1000000*5) = 5000008 bytes

2. ¿Cuánta memoria ocupa la `audioteca` (expréselo en MiBytes)?

Tamaño struct PISTA = Tamaño char[20] + Tamaño short + tamaño de CANAL[5] = (1 * 20) + 2 + (5000008 * 5) = 25000062 bytes

Tamaño audioteca[50] = 50*Tamaño PISTA = 50 * 25000062 = 1250003100 bytes

Pasar a MyBytes:

1250003100 bytes / 2^{20} = 1250003100 bytes / 1024^2 = 1192,1 MiBytes

3. Dado que el vector `audioteca` comienza en la dirección 0x4B0C8, ¿en qué dirección de memoria (expresada en hexadecimal) empieza el elemento `audioteca[25].canales[3].muestra[700]`?

Desplazamiento hasta `audioteca[25]` (desde el inicio de `audioteca`) = 25* tamaño `struct PISTA` = 25*25000062 Byte = 625001550 Byte

Desplazamiento hasta campo `canales` (desde el comienzo de la estructura `PISTA`) = tamaño `char[20]` +tamaño `short` = (1*20+2) Byte = 22 Byte

Desplazamiento hasta `canales[3]` (desde el comienzo de `canales`) = 3 * tamaño `CANAL` = 3 *5000008 Byte = 15000024 Byte

Desplazamiento hasta campo `muestra` (desde el comienzo de la estructura `CANAL`) = tamaño `int` +tamaño `int` = (4+4) Byte = 8 Byte

Desplazamiento hasta `muestra[700]` (desde el comienzo de `muestra`) = 700 * tamaño `MUESTREO` = 700*5 Byte = 3500 Byte

Desplazamiento total = (625001550 + 22 + 15000024 + 8 + 3500) Byte = 640005104 byte

Se pasa el resultado decimal a hexa:

$(640005104)_{10} = (2625B3F0)_{16}$

Dirección = dirección inicial + desplazamiento total = 4B0C8+ 2625B3F0 = 262A64B8

Se tienen las siguientes declaraciones en C:

```
struct INTERNO {
  char nombre[40];
  char nTratantes;
  short tratante[10];
  int cuarto;
};

struct DOCTOR {
  char nombre[40];
  short nPaciente;
  struct INTERNO paciente[20];
  int oficina;
};

struct HOSPITAL {
  char nombre[20];
  struct DOCTOR planta[150];
}  unHospital;
```

1. ¿Cuánta memoria ocupa un `DOCTOR` (expréselo en Bytes)?
2. ¿Cuánta memoria ocupa `unHospital` (expréselo en KiBytes)?
3. Dado que la estructura `unHospital` comienza en la dirección 0x7D55C, ¿en qué dirección de memoria (expresada en hexadecimal) empieza el elemento `unHospital.planta[20].paciente[15]. nTratantes`?

Se tienen las siguientes declaraciones en C:

```
struct CURSO {
  char programa[4];
  int materia;
  short nota[10];
  int semestre[10];
};

struct CODIGO {
  short anio;
  char sem;
  int secuencia;
};

struct ESTUDIANTE {
  struct CODIGO carne;
  char nombre[30];
  struct CURSO cursos[100];
};

struct ESTUDIANTE estudiantes[1000];
```

1. ¿Cuánta memoria ocupa un `CURSO` (expréselo en Bytes)?

Tamaño struct CURSO = tamaño char[4] + tamaño int + tamaño short[10] + tamaño int[10] = (4 + 4 + 2*10 + 4*10) byte = 68 Byte

2. ¿Cuánta memoria ocupa el vector `estudiantes` (expréselo en MiBytes)?

Tamaño struct CODIGO = tamaño short + tamaño char + tamaño int = (2 + 1 + 4) byte = 7 Byte

Tamaño struct ESTUDIANTE = tamaño struct CODIGO + tamaño char[30] + Tamaño struct CURSO[100] = 7 + 1*30 + 68*100 = 6837 Byte

Tamaño estudiantes[1000] = 1000*Tamaño struct ESTUDIANTE = 1000*6837 bytes= 6837000 Byte

Pasar a MiBytes:

6837000 bytes / 2^{20} = 6837000 bytes / 1024^2 = 6,52 MiBytes

3. Dado que el vector `estudiantes` comienza en la dirección 0x9A24C, ¿en qué dirección de memoria (expresada en hexadecimal) empieza el elemento `estudiantes [25].cursos[40].nota[3]`?

Desplazamiento hasta estudiantes[25] (desde el inicio de estudiantes) = 25* tamaño struct ESTUDIANTE = 25*6837 Byte = 170925 Byte

Desplazamiento hasta campo cursos (desde el comienzo de la estructura ESTUDIANTE) = tamaño struct CODIGO + tamaño char[30] = 7 + 30 Byte = 37 Byte

Desplazamiento hasta cursos[40] (desde el comienzo de cursos) = 40*tamaño struct CURSO = 40*68 Byte = 2720 Byte

Desplazamiento hasta campo nota (desde el comienzo de la estructura CURSO) = tamaño char[4] +tamaño int = (4+4) Byte = 8 Byte

Desplazamiento hasta nota[3] (desde el comienzo de nota) = 3 * tamaño short = 3*2 Byte = 6 Byte

Desplazamiento total = (170925 + 37 + 2720 + 8 + 6) Byte = 173696 Byte

Se pasa el resultado decimal a hexa:

$(173696)_{10} = (2A680)_{16}$

Nota:

```
173696 | 16
     0  10856 | 16
            8   678 | 16
                6    42 | 16
                     10  2
```

Dirección en hexa = dirección inicial + desplazamiento total = 9A24C+ 2A680 = C48CC

```
struct HORAS {
  int hora;
  char persona[40];
  char concepto[100];
};

struct DIA {
  char nombre[9];
  struct HORAS citas[50];
```

```
};
struct MES {
  char nombre[10];
  short numeroDias;
  struct DIA dias[31];
};
struct MES agenda[12];
```

1. ¿Cuánta memoria ocupa `HORAS` (expréselo en Bytes)?

Tamaño `struct HORAS` = tamaño `int` + tamaño `char[40]` + tamaño `char[100]` = (4+40+100) byte = 144 byte

2. ¿Cuánta memoria ocupa la `agenda` (expréselo en MiBytes)?

Tamaño `struct DIA` = tamaño `char[9]` + tamaño struct `HORAS [50]` = (9+50*144) byte = 7209 byte

Tamaño `struct MES` = tamaño `char[10]` + tamaño `short` + tamaño `struct DIA[31]` = (10+2+31*7209) byte = 223491 byte

Tamaño `agenda[12]` = 12*Tamaño `struct MES` = 12 *223491 byte= 2681892 byte

Pasar a Mibyte:

2681892 byte/ (2^{20} byte/ MiB) = 2,56 MiB

3. Dado que el vector `agenda` comienza en la dirección 0xA517D, ¿en qué dirección de memoria (expresada en hexadecimal) empieza el elemento `agenda[10].dias[25].citas[45].persona[20]`?

Desplazamiento hasta `agenda[10]` (desde el inicio de `agenda`) = 10* tamaño `struct MES` = 10*223491 byte = 2234910 byte

Desplazamiento hasta campo `dias` (desde el comienzo de la estructura `MES`) = tamaño `char[10]` +tamaño `short` = (10*1+2) Byte = 12 byte

Desplazamiento hasta `dias[25]` (desde el comienzo de `dias`) = 25 * tamaño `struct DIA` = 25 *7209 Byte = 180225 byte

Desplazamiento hasta campo `citas` (desde el comienzo de la estructura `DIA`) = tamaño `char[9]` = 9 byte

Desplazamiento hasta `citas[45]` (desde el comienzo de `citas`) = 45 * tamaño `HORAS` = 45*144 byte = 6480 byte

Desplazamiento hasta campo `persona` (desde el comienzo de la estructura `HORAS`) = tamaño `int` = 4 byte

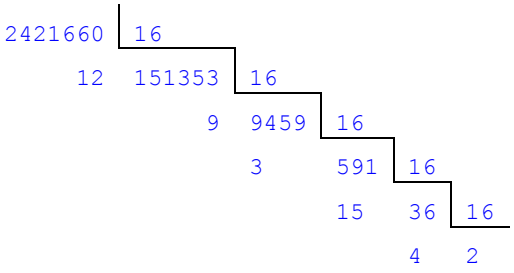
Desplazamiento hasta `persona[20]` (desde el comienzo de `persona`) = 20 * tamaño `char` = 20*1 byte = 20 byte

Desplazamiento total = (2234910 + 12 + 180225 + 9 + 6480 + 4 +20) byte = 2421660 byte

Se pasa el resultado decimal a hexa:

$(2421660)_{10} = (24F39C)_{16}$

Nota:



Dirección en hexa = dirección inicial + desplazamiento total = A517D+ 24F39C = 2F4519