Systèmes Embarqués 1 & 2

Classes T-2/I-2 // 2018-2019

a.12 – Modes d'adressage

Exercices

Exercice 1

Transcrire en assembleur ARM les algorithmes ci-dessous (l'usage des registres est fortement recommandé)

(a)

```
int i = 25;
int j = -10;
int k = i;
int l = i - j;
```

(b)

```
int i = 0xabefab98;
int j = 0;
for (int k = 31; k >= 0; k--)
   j += i & (1<<k) != 0 ? 1 : 0;
```

(c)

```
int i = -58909;
unsigned int j = 5;
int k = i >> j;       // i / (2**j)
int l = i / 128;
```

(d)

```
unsigned int i = 58909;
unsigned int j = 5;
unsigned int k = i / (1 << j);
unsigned int l = i / 128;
```

(e)

```
int i = -879;
unsigned int j = 6;
int k = i * (1 << j);
int l = i * 64;
```

(f)

```
unsigned int i = 879;
unsigned int j = 6;
unsigned int k = i * (1 << j);
unsigned int l = i * 64;
```

Exercice 2

Pour les codes assembleur effectuant des échanges de données avec la mémoire, la représentation de la mémoire (little-endian / 8-bits) et l'état des registres du processeur ci-dessous, indiquer le résultat des opérations

(a)

```
ldrb    r1, [r0]
ldrh    r2, [r0]
ldr     r3, [r0]

str     r1, [r0, #8]
strh    r2, [r0, #12]
strb    r3, [r0, #14]
```

(b)

```
ldrsb   r4, [r0, #4]!
ldrsh   r5, [r0], #2
ldr     r6, [r0, -r7]!

ldr     r10, [r0, r9, lsl #2]
str     r10, [r8, r9, lsl #2]
strb    r10, [r8, r9, lsl #1]
```

(c)

```
ldr     r10, [r0, r9, lsl #2]!
str     r10, [r8, r9, lsl #2]!
strb    r10, [r8, r9, lsl #2]!

ldrd    r4, [r11, #0]
strd    r4, [r0, #8]
```

Mémoire
(little-endian / 8 bits)

0xa0000100	0x25
0xa0000101	0x83
0xa0000102	0x75
0xa0000103	0x84
0xa0000104	0x87
0xa0000105	0x25
0xa0000106	0x73
0xa0000107	0xc2
0xa0000108	
0xa0000109	
0xa000010a	
0xa000010b	
0xa000010c	
0xa000010d	
0xa000010e	
0xa000010f	
0xa0000110	
0xa0000111	
0xa0000112	
0xa0000113	
0xa0000114	
0xa0000115	
0xa0000116	
0xa0000117	

Registres

R0	0xa0000100
R1	
R2	
R3	
R4	
R5	
R6	
R7	0x00000006
R8	0xa0000110
R9	0x00000001
R10	
R11	0xa0000100
R12	

Exercice 3

Transcrire en assembleur le code C ci-dessous effectuant des échanges de données avec la mémoire

(a)
```
struct S {long a ; long b; long c; long d;}
struct S s1;
struct S s2[10];
s1.a = 1; s1.b = 2; s1.c = 3; s1.d = 4;
for (int i=0; i<10; i++) {
  s2[i].a = 10;  s2[i].b = 10;  s2[i].c = 10;  s2[i].d = 10;
}
```

(b)
```
long l[10]; long * lp = &l[0];
for (int j=0; j<10; j++) *lp++ = j;
```

(c)
```
char src[10+1] = "0123456789";
char dst[10+1];
char* s = src;
char* d = dst;
while(*d++ = *s++);
```

(d)
```
char src[10+1] = "0123456789";
char dst[10+1];
register char* s = &src[11];
register char* d = &dst[11];
register int i;
for (i=11; i>0; i--)
  *--d = *--s;
```

(e)
```
unsigned char array[20];
unsigned long sum = 0;
for (int i=0; i<20; i++)
  sum += array[i];
```

Exercice 4

Echange de données avec la mémoire

(a) Sauver les registres R0, R3 et R6 à R10 sur une pile « full descending ». La pile est adressée avec le registre R12, lequel contient l'adresse 0xa0008000. Indiquer l'état de la pile et le contenu des registres du processeur qui ont été modifiés. Donner le code assembleur.

(b) Restituer d'une pile « full descending » les registres R1, R3, R5 et R7 à R10. La pile est adressée avec le registre R12, lequel contient l'adresse 0xa0007f00. Indiquer l'état de la pile et le contenu des registres du processeur qui ont été modifiés. Donner le code assembleur.