CSE 312: Microprocessor Based Systems

Section 4

Contact Information

tasneem.awaad@eng.asu.edu.eg

Coursework

Midterm	20 marks
Project	10 marks
2 Quizzes	10 marks

Addressing Modes

- MOV
  - Format: **MOV Rn, Op2**
  - **Op2** can be register or #immediate
  - MOV R0,#0x25
- LDR Rn, =const to move any 32-bit value into a register
  -

Addressing Modes

- Load/Store memory
  - Register indirect Addressing Mode
  - PC relative Addressing Mode
  - PUSH and POP Register Addressing Mode

Addressing Modes

- Load/Store memory
  - Register indirect Addressing Mode
    - **Regular Register indirect**
      - LDR R7, [R5]
      - R5 unchanged, R7 = Mem[R5]
    - **With Immediate Offset:**
      - LDR R7, [R5, #4]
      - R5 unchanged, R7 = Mem[R5 + 4]
    - **With Register Offset:**
      - LDR R7, [R5, R4]
      - R5 Unchanged, R7 = Mem[R5 + R4]

Addressing Modes

- Load/Store memory
  - Register indirect Addressing Mode
    - **With Pre indexed Immediate Offset:**
      - LDR R7, [R5, #4]!
      - R5 = R5 + 4
      - R7 = Mem[R5]
    - **With Post indexed Immediate Offset:**
      - LDR R7, [R5], #4
      - R7 = Mem[R5]
      - R5 = R5 + 4
    - **With Shifted Register Offset:**
      - LDR R7, [R5, R4, LSL #2]
      - R5 Unchanged, R7 = Mem[R5 + R4<<2]

Sheet 3

- Write a complete ARM assembly program for the procedure func2. The procedure func2 calculates this C expression ((X+Y)>>3) – Z and stores its value in R0. Assume X, Y, Z are 32-bit signed numbers. X, Y, Z are defined in the memory as shown

```
        AREA   mydata, DATA, READONLY
X        DCD    -20
Y        DCD    -60
Z        DCD    -20
```

Answer

```
func2
        LDR R0, =X          ; load address of X into R0
        LDR R1, =Y
        LDR R2, =Z
        LDR R3, [R0]         ; R3= -20
        LDR R4, [R1]         ; R4= -60
        ADD R0, R3, R4       ; R0=X+Y --> -20 + -60
        ASR R0, R0, #3       ; (X+Y) >> 3
        LDR R5, [R2]         ; R5= -20
        SUB R0, R0, R5       ; ((X+Y) >> 3) -Z
        BX LR
```

Sheet 3

- Translate the below C code into ARM assembly code, using the registers indicated by the variable names. The C code presumes that r0 holds the address of the first entry of an array of integer values, and r1 indicates how many elements the array holds; the code removes all adjacent duplicates from the array.

```
r3 = 1;
for (r2 = 1; r2 < r1; r2++) {
    if (r0[r2] != r0[r2 - 1]) {
        r0[r3] = r0[r2];
        r3 += 1;
    }
}
r1 = r3;
```

Answer

```
            MOV R3, #1                   ; counter for nonduplicated items
            MOV R2, #1                   ; loop iterator
            MOV R1, #Array_Size
LOOP1       CMP R2, R1                   ; R2- R1
            BGE Done                     ; R2> R1 jump to Done
            LDR R4, [R0,R2, LSL #2 ]     ; R4= MEM[R0+R2*4]
            SUB R5, R2, #1               ; R5=R2-1
            LDR R5, [R0, R5, LSL #2]     ; R5= MEM[R0 + (R2-1)*4]
            CMP R4, R5                   ; R0[R2]!= R0[R2-1] --> R0[i]!= R0[i-1]
            STRNE R4, [R0,R3, LSL #2 ]   ; R0[R3]= R0[R2], Z flag=0 N flag
            ADDNE R3, R3, #1             ; R3=R3+1
            ADD   R2, R2, #1             ; R2=R2+1
            B LOOP1

Done       MOV R1, R3
```

Sheet 3

- Translate the below C fragment into an equivalent ARM assembly language program, using registers corresponding to the variable names. Assume r0 and r1 hold signed values.

```
r2 = 0;
while (r1 != 0) {
    if ((r1 & 1) != 0) {
        r2 += r0;
    }
    r0 <<= 1;
    r1 >>= 1;
}
while (1); // halting loop
```

Answer

```
            MOV R2, #0
LOOP_2      CMP R1, #0
            BEQ Halt_LOOP
            TST   R1, #1
            ADDNE R2, R2, R0
            LSL   R0, R0, #1
            ASR   R1, R1, #1
            B LOOP_2

Halt_LOOP B Halt_LOOP
```

Sheet 3

- In a digital clock embedded system, you need to implement a function that lets a user pressing a button to display the day of the year. Write an Embedded C function that takes 3 parameters of the day, month and year, performs proper checks on all inputs and returns the day of year (1 – 366). Leap year is that divisible by 4 and 400 but not divisible by 100.

Answer

```c
#include <stdint.h>

static char daytab[2][13]={
{0,31,28,31,30,31,30,31,31,30,31,30,31},
{0,31,29,31,30,31,30,31,31,30,31,30,31}
};

static int now=0;

int day_of_year(int year,int month,int day)
{
int i,leap;

    leap=  (year%4==0 && year%100 !=0) || year %400 == 0;


if(month < 1 || month > 12)
        return -1;

if(day < 1 || day > daytab[leap][month])
        return -1;

for(i=1;i<month;i++)
day += daytab[leap][i];
return day;
}

int main(){
int day=20;
int year=2020;
int month=4;
now=day_of_year(year,month,day);

return 0;
}
```