For: Dr. Le’s Fall 24 Class

Microprocessor Exam 1 Practice Questions

Contents

Numbers

Changing Bases

1) What is the difference between signed and unsigned?
2) Given a signed integer represented using two's complement notation with *n* bits, what is the range of possible values that can be represented? Provide the general formula and an example for an 8-bit signed integer?
   a) What is the range of values that can be represented by an 8-bit signed integer using two's complement notation?
      i) A 16-bit signed integer?
      ii) A 16-bit unsigned integer?

Binary to Decimal

3) Find the decimal equivalent of the following signed binary values:
   i) 101010_2
   ii) 111001_2
   iii) 100111_2
   iv) 110000_2
   v) 011000.111_2
   vi) 10110101.011_2
   vii) 1100101001111101.0111_2
   viii) 1010111100000101.1011_2

Decimal to Binary

4) Find the binary equivalent of the following decimals:
   i) 12.5_{10}
   ii) 112.97_{10}
   iii) 1804.864_{10}
   iv) 12508.123_{10}
   v) 761.30_{10}
   vi) 46572.152_{10}

Decimal to Hex & Octal

5) Find the Hex & Octal equivalent of the following decimals:
   i) 14376.16_{10}
   ii) 8999.0625_{10}
   iii) 26078.375_{10}
   iv) 53027.011_{10}
   v) 12345.648_{10}

Hex & Octal to Decimal

6) Find the Decimal equivalent from the following signed Hex/Octal:

i) $A2C_{16}$
ii) 3408_{16}
iii) $FFFF\_FFFF_{16}$
iv) 120_8
v) $AB8F\_E327_{16}$
vi) $00A0\_241B_{16}$

Arithmetic/Shift/Rotate Operations Condition Flags

Arithmetic Condition Flags

7) In arithmetic, how do you “activate” the flags in Assembly?

8) What condition flags are affected by a signed addition?

9) Unsigned Subtraction?

10) When is C set? Cleared?

11) When is V set?

12) When is N set?

13) When is Z set?

14) What makes an unsigned addition “bad”?
    i) What becomes set or cleared if unsigned addition is bad?
    ii) Give me an example of bad unsigned addition using money.

15) What makes unsigned subtraction “bad”?
    i) What becomes set or cleared if unsigned subtraction is bad?
    ii) Give me an example of bad unsigned subtraction using money.

16) What makes signed addition “bad”?
    i) What becomes set or cleared when signed addition is bad?
    ii) What becomes set or cleared when signed addition is good?

17) What makes signed subtraction “bad”?
    i) What becomes set or cleared when signed addition is bad?
    ii) Good?

18) How does unsigned arithmetic relate to each other regarding conditionals? Signed arithmetic?

19) What condition flags are set given the following arithmetic operations:
   i) 0x7FFF_FFFF + 0x2
   ii) 0x7FFF_FFFF – 0x200
   iii) 0x100 + 0x500
   iv) 0x100 – 0x300
   v) 0x7FFF_FFF + 0x1
   vi) 0X7FFF_FFF + 0x1

Shift/Rotate Operations with Condition Flags

20) What flag(s) does the shift and rotate operation set/clear (LSL,ASR,LSR,ROR, RRX)

21) When should you use Arithmetic shift right (ASR) versus logical shift right (LSR) and why?

22) Write an ARM assembly code that performs a LSL on the value 0x40000000 by 2 bits. What is the result, and what condition flags are set after the operation?

23) Given the signed number -4 (0Xfffffffc), write ARM assembly code that performs an Arithmetic Shift Right by 1 bit. What is the result, and how are the condition flags affected?

24) Write an ARM assembly code to perform a Rotate Right with Extend (RRX) on 0x80000000. What is the result if the carry flag is initially set to 1?

25) Write an ARM assembly code that performs a Rotate Right (ROR) on the value 0x00000003 by 1 bit. What is the result, and what happens to the carry flag?

26) Write an ARM assembly code that performs a LSL on the value 0x40000000 by 1 bit. What is the result and the condition flags (N,Z,C)?

27) Write an ARM assembly code that multiples the value 0x10000000 by 4 using shifts or rotates only.

28) Write an ARM assembly code that divides the value of 0x80000000 by 8 using shifts or rotates only.

Branching

29) What is the format used to write a branch statement? There are 2 answers, 1 is used more than the other.

30) Write an assembly code snippet that checks if a number stored in a register is positive. If it is positive, branch to a label that adds 5 to the number; otherwise, branch to a label that subtracts 5 from the number.

31) Write an assembly code snippet that checks if a number stored in a register is less than 10. If it is less than 10, branch into a subroutine, otherwise branch to a different subroutine.

32) Convert the following C code to Assembly:

```
uint8_t B;
if(B != 0) {
   Bnotzero();
} else {
   Bequalzero();
}
```

33) Convert the following C program to Assembly:

```
if (A < -20)
   isTooLow();
else if (A > 30)
   isTooHigh();
else
   isWithinRange();
```

34) Convert the following C code to assembly:

```
if (B == 0 && C < 0)
   isBZeroAndCNegative();
else if (B > 0 || C > 0)
   isEitherPositive();
else
   isBothNegativeOrZero();
```

Assume there is a 32-bit signed number G. Write an assembly program that implements the following C code: if (G>100) function1() elseif (G<50) function2() else function3()

```
        //RAM//
        //ROM//
        ALIGN
        Export Start
Start
        LDR R2, =G ;R2 = &G
        LDR R0, [R2] ;R0 = G
        CMP R0, #100
        BGT hi100
        CMP R0, #50
        BLT hi50
        BL function3
        B next
hi50    BL function2
```

```
                B next
        hi100   BL function1
        next
```

Assume there is a 16-bit signed number G. Write an assembly program that implements the following code: while(G>0) funcA{}

```
        //RAM// AREA Myvariables, DATA, READWRITE, Align = 2
varG    space 2
        //ROM// AREA |.text|, code, READONLY, Align = 2
        ALIGN
        Export Start
Start
        LDR R4, =varG
loop    LDR R0, [R4]
        CMP R0, #0
        BLE next
        BL funcA
        B loop
next
```

Assume there is a 32-bit signed number K. Write an assembly program that implements the code: {if(K<100) and K>10} function();

```
        //RAM//
varK    space 4
        //ROM//
        ALIGN
        Export Start
Start
        LDR R2, =varK
        LDR R0, [R2]
        CMP R0, #100
        BGE hi100
        BLE lo10
hi100   B next
lo10    B next
        BL function
next
```

ASCII Code

35) Decode the following binary ASCII codes:

i) 1001000_1100101_1101100_1101100_1101111_0100000_1010111_1101111_1110010_1101100_1100100

ii) 1010100_1100100_1100101_1110011_1101110_1110101_1101011_1101011_1101101_1100101_1101110

iii) 1001110_1100101_1101111_1101110_0100000_1001100_1101111_1110110_1100101_0100000_1100101_1101001_1110100

Push and Pop

36) Show the contents of the stack after the three marked points in the execution of the following code. Assume: R0 = 0x30, R1 = 0x31, R2 = 0x32, R3 = 0x33, R4 = 0x34, R5 = 0x35, R6 = 0x36. The initial stack pointer (SP_0 = 0x20000200).

```
PUSH {R2, R3, R0, R1}
ADD R3, R1, R0
POP {R1, R2, R4}  <---- A
ADD R5, R1, R2
PUSH {R0, R5, R4, R3}  <---- B
POP {R2, R1, R0, R4}  <---- C
```

37) Show the contents of the stack after the three marked points in the execution of the following code. Assume: R0 = 0x40, R1 = 0x41, R2 = 0x42, R3 = 0x43, R4 = 0x44, R5 = 0x45, R6 = 0x46. The initial stack pointer (SP_0 = 0x20000300.)

```
PUSH {R1, R0, R2, R4}
ADD R5, R1, R0
POP {R4, R1, R3}  <---- A
ADD R6, R3, R1
PUSH {R2, R5, R6, R0}  <---- B
POP {R0, R4, R2, R3}  <---- C
```

38) Show the contents of the stack after the three marked points in the execution of the following code. Assume: R0 = 0x50, R1 = 0x51, R2 = 0x52, R3 = 0x53, R4 = 0x54, R5 = 0x55, R6 = 0x56. The initial stack pointer (SP_0 = 0x20000400.)

```
PUSH {R0, R6, R4, R1}
ADD R3, R1, R0
POP {R2, R3, R5}  <---- A
ADD R4, R2, R3
PUSH {R5, R0, R2, R6}  <---- B
POP {R4, R3, R1, R0}  <---- C
```

Answers

Changing Bases

1) Signed numbers include both positive and negative values while unsigned are only positive values. The range of signed values are from -2^7 to 2^7-1 and the range of unsigned values are from 0 to 2^8-1.
2) -2^n to 2^n-1.
    b) -127 to 128
        i) -32,768 to 32,767
        ii) 0 to 65,535

Binary to Decimal

i) 42_{10}
ii) 65_{10}
iii) 39_{10}
iv) 48_{10}
v) 24.875_{10}
vi) -75.375_{10}
vii) $-13{,}699.4375_{10}$
viii) $-20{,}731.6875_{10}$

Decimal to Binary

i) 1100.1_2
ii) 1110000.1111100001_2
iii) $11100001100.11011101001_2$
iv) $11000011011100.011111011111_2$
v) $1011111001.010011001100110_2$
vi) $011010111101100.001001101110_2$

Decimal to Hex & Octal

i) $3828.28F5C_{16}$ & 34050.12172024_8
ii) 2327.1_{16} & 21447.04_8
iii) $65DE.6_{16}$ & 62736.3_8
iv) $CF23.02D0E56_{16}$ & 147443.00550345_8
v) $3039.A5D353F_{16}$ & 30071.51361523_8

Hex & Octal to Decimal

i) 2604_{10}
ii) 13320_{10}

iii) -1_{10}
iv) 80_{10}
v) -1416633561_{10}
vi) 10495003_{10}

Condition Flags

7) By adding the suffix “S” to arithmetic operation, the flags are enabled and can be seen in the PSR (Program Status Register)
8) For signed numbers, the overflow V bit is set or cleared.
9) For unsigned numbers, the carry C bit is set or cleared
10) C is set when unsigned addition crosses the 0 boundary OR when unsigned subtraction is a “good” result (I.e 0x10-0x2 = 0xE). C is cleared when unsigned addition is a “good” result OR when unsigned subtraction crosses the 0 boundary.
11) V is set when a signed result crosses the 0x8000_0000/0x7FFF_FFFF boundary. V is cleared when a signed result does not cross the previously stated boundary AND the result is correct.
12) N is set when the result is negative. N is cleared when the result is NOT negative.
13) Z is set ONLY when the result is 0. Otherwise, Z is cleared.
14) Crossing the 0 boundary makes unsigned addition bad.
i) The C bit is set when unsigned addition is bad.
ii) You have $999 in the bank. You deposit $200 and your new running total balance is $199. (You crossed the “0” and your addition makes no sense. You should have $1199)
15) Just like unsigned addition, crossing the 0 boundary makes unsigned subtraction bad.
i) The C bit is cleared when unsigned subtraction is bad.
ii) You have $100 in your bank account and want to withdraw $300. The bank makes a mistake and gives you the money and your new running total balance is $800. (You crossed the “0” and now the subtraction makes no sense.)
16) Crossing the 0x8000_0000/0x7FFF_FFFF boundary makes signed addition bad.
i) V is set when signed addition is bad.
ii) V is cleared when signed addition is good.
17) Crossing the 0x8000_0000/0x7FFF_FFFF boundary makes signed addition bad.
i) V is set when signed addition is bad.
ii) V is cleared when signed addition is good.
18) Unsigned arithmetic is opposite for flags. (C=0 or C=1) And Signed arithmetic has the same, no matter addition or subtraction.
19i) N and V
19ii) C
19iii) NONE
19iv) N
19v) C
19vi) N and V

Shift/Rotate Operations with Condition Flags

20) The C flag.
21) ASR is used for signed numbers to preserve the sign (MSB). LSR is used for unsigned numbers since the MSB doesn’t matter.
22)

```
        MOV R0, #0x40000000
        LSLS R0, R0, #2
```

Result: 0x00000000
Condition Flags: N: 0, Z: 1, C: 1, V: 0

23)

```
        MOV R0, #-4
        ASRS R0, R0, #1
```

Result: 0xFFFFFFFE
Condition Flags: N:1, Z: 0, C: 0, V: 0

24)

```
        MOV R0, #0x80000000
        RRX R1, R0
```

Result: 0xC0000000 = $1100\_0000\_0000\_0000\_0000\_0000\_0000\_0000_2$
C is cleared

25)

```
        MOV R0, #0X00000003
        ROR R0, R0, #1
```

Result: 0x80000001
Carry is set

26)

```
        MOV R0, #0x40000000
        LSL R0, R0, #1
```

Result: 0x40000000
N: Set, C: Clear, Z: Clear

27)

```
        MOV R0, #0x10000000
        LSL R0, R0, #2
```

Result: 0x40000000

28)

```
        MOV R0, #0x80000000
        LSR R0, R0, #3
```

Result: 0x10000000

Branching

29)

B{cond} label ;Branch without planning to return.

BL{cond} label ;Branch that saves the return address, allowing a return after the function or subroutine call.

ASCII Code

35)
i) Hello World
ii) Tdesnukkmen
iii)Neon Love eit

PUSH POP

36) SP_A =
R0 =
SP_B =
R0 =
SP_c = 0x200001FC
R0 = 0X30, R1 = 0X61, R2 = 0X32, R3 = 0X61, R4 = 0X61, R5 = 0X61, R6 = 0X36

37) SP_A =
The stack has in order from top to bottom:
SP_B =
R0 =
SP_f = 0x200002FC
R0 = 0x40, R1 = 0X40, R2 = 0X41, R3 = 0X41, R4 = 0X42, R5 = 0X81, R6 = 0X81