



CS1021 Tutorial #5

More Pseudo-code and Flow Control

1 Translating Pseudo-code into ARM Assembly Language

Translate each of the following pseudo-code programs into ARM Assembly Language.

- (a) Assume `m` and `n` are signed values stored in R8 and R9 respectively

```
if (m >= 100 && n < 10)
{
    m = m + n;
}
```

- (b) Assume `x` and `y` are unsigned values stored in R2 and R3 respectively

```
if (x == 5 || x == 15)
{
    y = y + 1;
}
```

- (c) Assume `ch` is an ASCII character stored in R3 and `v` is an unsigned value stored in R1.

```
if (ch=='a' || ch=='e' || ch=='i' || ch=='o' || ch=='u')
{
    v = v + 1;
}
```

- (d) Assume `ch` is an ASCII character stored in R0 and `count` is an unsigned value stored in R1.

```
if ( (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z') )
{
    count = count + 1;
}
```



- (e) Assume `ch` is an ASCII character stored in `R6` and `a`, `b` and `c` are signed values stored in `R0`, `R7` and `R8` respectively.

```
if (ch == '+')
{
    a = b + c;
}
else if (ch == '-')
{
    a = b - c;
}
else if (ch == '*')
{
    a = b * c;
}
else
{
    a = 0;
}
```

2 Reading User Input

- (a) The pseudocode below illustrates an approach to read a sequence of characters entered by a user. The program stops reading input when the user presses the return key, which we will assume corresponds to the ASCII character code `0x0D`.

A single character is read by calling the `getchar()` method. Unfortunately, `getchar()` by itself does not display the key pressed by the user so we need to do this ourselves. The function `sendchar(ch)` displays the character with ASCII code `ch`.

```
// read first character
ch = getchar();

// continue reading until RETURN is pressed
while (ch != 0x0D)
{
    sendchar(ch);
    ch = getchar();
}
```

Modify the pseudo-code so it reads a sequence of digits (`'0' ... '9'`) representing a multi-digit decimal value and converts the sequence of digits into the value they represent. For example, if the user types `'1'`, `'2'`, `'4'`, the program should convert this to the value 124.

- (b) Convert your pseudo-code into ARM Assembly Language. Replace invocations of `getchar()` with the instruction `BL getchar` and assume that the character code will be stored in `R0`. Similarly, replace invocations of `sendchar()` with the instruction `BL sendchar` after storing the code for the character to be displayed in `R0`.