Lab 2

Instructions

Complete each task and demonstrate the working program to your tutor. Tasks should be demonstrated using AVR Studio's simulator. You will have two lab sessions to work on this lab, and all questions must be marked by the end of the second session.

Part A – Reverse String (2 Marks)

Implement a program that loads a null-terminated string from program memory and pushes it onto the stack, then writes out the reversed string into data memory. The stack pointer must be initialized before use.

Eg "abc",0 will be stored in RAM as "cba",0

Part B – Function Calls (4 Marks)

The following code calls a function named **checkalpha** twice to evaluate if two strings contains only alpha characters (a-z and A-Z).

```
.include "m2560def.inc"

.cseg
rjmp start

validstring:
.db "abcdABCD", 0
invalidstring:
.db "74(*&Q#$^}{:?<>", 0

start:
; !!! Insert stack initialization code here !!!

ldi ZL, low(validstring << 1)
ldi ZH, high(validstring << 1)
call checkalpha
mov r20, r16
; r20 should be 1

ldi ZL, low(invalidstring << 1)
ldi ZH, high(invalidstring << 1)
call checkalpha
mov r21, r16
; r21 should be 0

halt:
rjmp halt
```

(continued)

Complete the code by adding the stack initialization code and writing an implementation of **checkalpha**. The above code should not be modified apart from adding the stack initialization code.

checkalpha should take a pointer to the string in the Z register, and return in r16 either 1 (if the string was alpha-only) or 0 (if the string contained non-alpha characters).

The C signature for **checkalpha** would be:

```
int checkalpha(char *str);
```

Part C – Indirect Calls (4 Marks)

Implement a superior string-checking function named **checkstring** that can check all characters in the string match an arbitrary predicate, by calling a function to check each character. The function used for checking must be passed as a function pointer.

The C signature for **checkstring** would be:

```
int checkstring(char *str, function *predicate);
```
(where 'function *' is the address of a function).

An example predicate would be:

```
int isdigit(char c)
{
    return c >= '0' && c <= '9';
}
```

checkstring would be called using:

```
checkstring("12345678", &isdigit);
```

checkstring should take the pointer to the string in Z, and the pointer to the predicate function in Y. It should return 1 in r16 if all characters match the predicate, otherwise it should return 0 in r16.

To get full marks, this should be demonstrated with at least two different predicate functions, and several input strings.