DT228-1 Microprocessors In class test: 28/4/2014

Answers in Italic

(1) What are the normal uses of the following ARM Cortex registers?

PC,PSR,SP,LR [8]

PC = Program counter : points to next instructions
PSR = Program status register: contains status,control and arithmetic flags.
SP = Stack pointer: points to last value pushed on to stack
LR = Link Register : stores return address after a Branch With Link (BL) instructions

(2) A program defines a symbol 'Y' as follows:

Y DCD 0

What pair of ARM Cortex assembler (Thumb) instructions would you use to load the contents of memory identified by the symbol Y into register R0. [4]

LDR R1,=Y
LDR R0,[R1]

(3) How do you declare an array of memory 20 bytes long in the GNU ARM assembler [4]

Ary SPACE 20

(4) State, and explain the use of two ARM assembler directives [4]

end : stops the assembly process
Export: adds a symbol to the export table from a module allowing other modules to link to it.

(5) What ARM Cortex flags will be set by the following calculations:

2-2 *: Zero flag (Carry = Not Borrrow)*
0x7fff ffff + 1 *: Negative flag and Overflow flag*
0xffff ffff + 1 *: Carry Flag and Zero Flag*
0 -1 *: Negative flag* [8]

(6) On entry into a subroutine, the contents of an ARM Cortex processor registers are as follows:
SP=0x10002000
R0=0x12345678
LR=0x00112244

The first two instructions in the subroutine are:
push {LR}
push {R0}

List the address and contents of the stack after these two instructions (you may assume the stack was empty before these instructions were executed.
Note: This is a little endian processor. [8]

In byte terms:

00 (Higher address)
11
22
44
12
34
56
78 (Lower address)

(7) Complete the code in the memcpy subroutine outlined below:

```
; void memcpy(char * dst, char * src, unsigned len);
; This function takes three arguments:
; a source buffer (src) and a destination
; buffer (dst).  The function copies the
; contents of src to dst.  The third argument
; is the number of bytes to be copied.
; Arguments are passed as follows:
; R1 points to the destination buffer
; R2 points to the source buffer
; R3 contain a count of the number of bytes to be copied
memcpy
        push {LR,R0-R3}

Loop    CMP R3,#0
        BEQ Exit
        LDR R0,[R2]
        STR R0.[R1]
        ADDS R1,R1,#4
        ADDS R2.R2,#4
        SUBS R3,R3,#1
        B Loop
Exit
        pop {PC,R0-R3}
```

[8]