EEE3095/6S Term 3 Quiz

30 August 2023

Duration: 45 minutes

Total: 50 marks

Empl ID: ______________________

Instructions:

- This is a closed-book test.
- There is only one part to this test.
- You must use a pen to complete the test — pencil will only be marked for drawings. Values indicated on drawings must be written in pen. Please show all of your working clearly — method and approach matter.
- Keep all your answers within the allocated blocks; anything outside of these blocks may not be marked.
- A formula sheet appears at the end of this paper.

Question	Available Marks	Grade
1	10	
2	20	
3	20	
Total:	50	

Part A

Empl ID: ☐☐☐☐☐☐☐

Question:	1	2	3	Total
Points:	10	20	20	50
Score:				

Question 1: Multiple Choice [10 marks]

a. If a Serial Peripheral Interface (SPI) protocol is configured to operate in SPI Mode 2, the logic level of the clock signal is: (2)

- ◯ high while idling
- ◯ low while idling
- ◯ always high
- ◯ always alternating

b. The 32-bit value 0x13A04B71 needs to be stored in memory; if using Big Endian, which value would be stored at the smallest memory address? (2)

- ◯ 0b01110001
- ◯ 0b00100001
- ◯ 0b00010011
- ◯ 0b00010111

c. How would a dynamic, asynchronous real-time system function? (2)

- ◯ Predictable task arrival with periodic execution
- ◯ Predictable task arrival with aperiodic execution
- ◯ Unpredictable task arrival with periodic execution
- ◯ Unpredictable task arrival with aperiodic execution

d. An execution environment can have more than one runtime environment. True or false? (2)

- ◯ True
- ◯ False

e. An Application Binary Interface (ABI) is said to be at "source-code level" and relates to the interface between different software components. True or false? (2)

- ◯ True
- ◯ False

Question 2: ARM Processing [20 marks]

a. Define the terms **RISC** and **CISC** and explain the main difference between them. (3)

b. Two well-known optimisation strategies are Small Function In-lining (SFI) and Loop Unrolling (LUR). Briefly describe each of these with at least one advantage associated with the optimisation. (4)

c. Mixing ARM Assembly code with C code requires the use of the same Application Binary Interface (ABI), and this defines the calling convention for our functions. Answer the following questions:

(i) What is the difference between an ABI and an API (Application Programming Interface)? (2)

(ii) A calling convention can be split up into two components; name these parts and provide a brief description for each one. (2)

(iii) List two reasons why a programmer may want to combine ARM Assembly and C code. (2)

d. Given the following ARM Assembly code, what would be the base-10 value in register **r0** after each line of code? Briefly explain your working. (7)

```
mov r0, #16
mov r1, 0b1000
add r0, r0, r1
asr r0, #2
lsr r0, #1
mul r1, r0, r1
mov r0, 0x11
```

Question 3: Embedded Communications [20 marks]

a. The 7-bit data word 0b1010111 is to be transmitted with a parity bit appended; what would be the final 8-bit data word that is transmitted in both odd- and even-parity systems? (2)

b. SPI uses a Slave Select line to initiate communication with a specific slave device, whereas I2C (Inter-Integrated Circuit) uses a different means of selecting the slave device with which the master communicates. Explain how this is done, and describe what would happen if an error occurs during this slave selection process. (2)

c. RS-485 uses a "twisted cable" configuration; what is the purpose of this? (2)

d. The following questions relate to the SPI communication protocol.

(i) Figure 1 shows the basic structure of the SPI protocol's interface. Provide names for the four missing labels. (2)

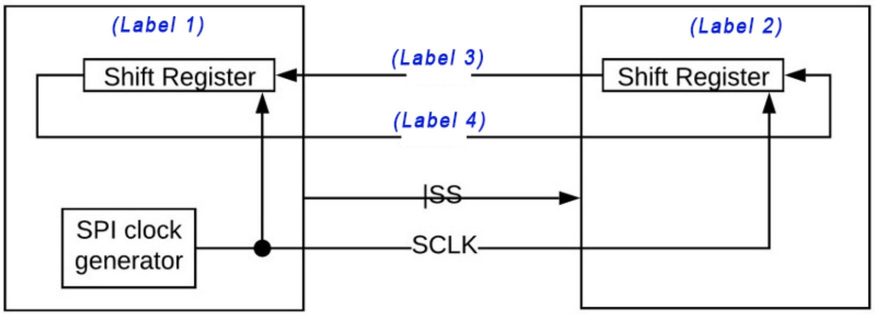

Figure 1: SPI interface with missing labels

(ii) SPI is said to be a full-duplex protocol. Explain what this means and make reference to Figure 1 in your answer. (2)

(iii) The four SPI Modes are dependent on the Clock Phase (CPHA) and Clock Polarity (CPOL). Explain these two terms. (2)

(iv) Based on the above, draw a basic SPI timing diagram to illustrate the differences between all four SPI Modes; you only need to draw the clock lines and SS. (4)

(v) Draw a diagram to illustrate how we could connect three **independent** slave devices to a single master device using SPI. (4)

END OF TEST

Appendix B: ARM Assembly Language Cheatsheet

Memory access instructions

```
LDR Rd, [Rn]             ; load 32-bit number at [Rn] to Rd
LDR Rd, [Rn,#off]        ; load 32-bit number at [Rn+off] to Rd
STR Rt, [Rn]             ; store 32-bit Rt to [Rn]
STR Rt, [Rn,#off]        ; store 32-bit Rt to [Rn+off]
PUSH {Rt}                ; push 32-bit Rt onto stack
POP  {Rd}                ; pop 32-bit number from stack into Rd

MOV{S} Rd, <op2>         ; set Rd equal to op2
MOV Rd, #im16            ; set Rd equal to im16, im16 is 0 to 65535
MVN{S} Rd, <op2>         ; set Rd equal to -op2
```

Branch instructions

```
B label                  ; branch to label Always
BEQ label                ; branch if Z == 1 Equal
BNE label                ; branch if Z == 0 Not equal
BCS label                ; branch if C == 1 Higher or same, unsigned ≥
BHS label                ; branch if C == 1 Higher or same, unsigned ≥
BCC label                ; branch if C == 0 Lower, unsigned <
BLO label                ; branch if C == 0 Lower, unsigned <
BMI label                ; branch if N == 1 Negative
BPL label                ; branch if N == 0 Positive or zero
BVS label                ; branch if V == 1 Overflow
BVC label                ; branch if V == 0 No overflow
BHI label                ; branch if C==1 and Z==0 Higher, unsigned >
BLS label                ; branch if C==0 or Z==1 Lower or same, unsigned ≤
BGE label                ; branch if N == V Greater than or equal, signed ≥
BLT label                ; branch if N != V Less than, signed <
BGT label                ; branch if Z==0 and N==V Greater than, signed >
BLE label                ; branch if Z==1 or N!=V Less than or equal, signed ≤
BX Rm                    ; branch indirect to location specified by Rm
BL label                 ; branch to subroutine at label
BLX Rm                   ; branch to subroutine indirect specified by Rm
```

Logical instructions

```
AND{S} {Rd,} Rn, <op2>   ; Rd=Rn&op2 (op2 is 32 bits)
ORR{S} {Rd,} Rn, <op2>   ; Rd=Rn|op2 (op2 is 32 bits)
EOR{S} {Rd,} Rn, <op2>   ; Rd=Rn^op2 (op2 is 32 bits)
BIC{S} {Rd,} Rn, <op2>   ; Rd=Rn&(~op2) (op2 is 32 bits)
ORN{S} {Rd,} Rn, <op2>   ; Rd=Rn|(~op2) (op2 is 32 bits)
LSR{S} Rd, Rm, Rs        ; logical shift right Rd=Rm>>Rs (unsigned)
LSR{S} Rd, Rm, #n        ; logical shift right Rd=Rm>>n (unsigned)
ASR{S} Rd, Rm, Rs        ; arithmetic shift right Rd=Rm>>Rs (signed)
ASR{S} Rd, Rm, #n        ; arithmetic shift right Rd=Rm>>n (signed)
LSL{S} Rd, Rm, Rs        ; shift left Rd=Rm<<Rs (signed, unsigned)
LSL{S} Rd, Rm, #n        ; shift left Rd=Rm<<n (signed, unsigned)
```

Arithmetic instructions

```
ADD{S} {Rd,} Rn, <op2>    ; Rd = Rn + op2
ADD{S} {Rd,} Rn, #im12    ; Rd = Rn + im12, im12 is 0 to 4095
SUB{S} {Rd,} Rn, <op2>    ; Rd = Rn - op2
SUB{S} {Rd,} Rn, #im12    ; Rd = Rn - im12, im12 is 0 to 4095
RSB{S} {Rd,} Rn, <op2>    ; Rd = op2 - Rn
RSB{S} {Rd,} Rn, #im12    ; Rd = im12 – Rn
CMP Rn, <op2>             ; Rn – op2 sets the NZVC bits
MUL{S} {Rd,} Rn, Rm       ; Rd = Rn * Rm signed or unsigned
```

Notes

```
Ra Rd Rm Rn Rt represent 32-bit registers
value any 32-bit value: signed, unsigned, or address
{S} if S is present, instruction will set condition codes
#im12 any value from 0 to 4095
#im16 any value from 0 to 65535
{Rd,} if Rd is present Rd is destination, otherwise Rn
#n any value from 0 to 31
#off any value from -255 to 4095
label any address within the ROM of the microcontroller
op2 the value generated by <op2>
```