NANYANG TECHNOLOGICAL UNIVERSITY

SEMESTER 1 EXAMINATION 2014-2015

CZ1006 – COMPUTER ORGANIZATION AND ARCHITECTURE

Nov/Dec 2014 Time Allowed: 2 hours

INSTRUCTIONS

1. This paper contains 4 questions and comprises 7 pages.
2. Answer **ALL** questions.
3. This is a closed-book examination.
4. All questions carry equal marks.
5. The VIP Instruction Set Summary Chart is provided in Appendix 1 on page 7.

---

1. Figure Q1a shows the hexadecimal contents of several registers in the VIP processor and a section of its memory.

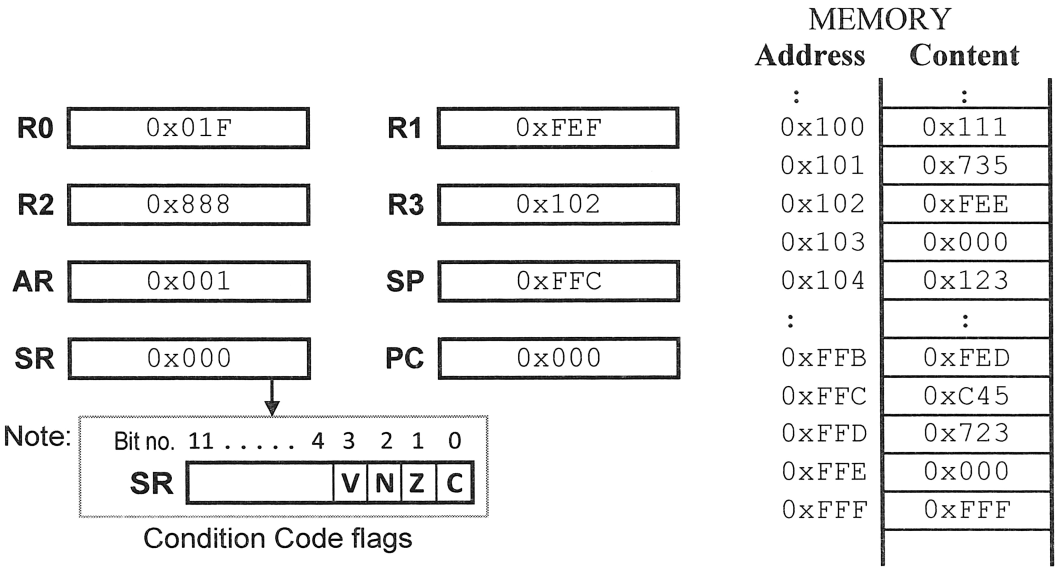

Figure Q1a

Note: Question No. 1 continues on Page 2

(a) Give (*in hexadecimal*) the 12-bit contents in the two registers **R0** and **SR**, immediately after the execution of each instruction given below.

Note: Instructions (i) to (vi) are **not consecutive instructions**. You must use the initial conditions shown in Figure Q1a to derive your answer for each of the instructions given below.

(i) `MOV R0,R2`

(ii) `MOV R0,[0x103]`

(iii) `EOR R0,[R3+0xFFE]`

(iv) `POPM 0x003`

(v) `ADD R0,#0x7F1`

(vi) `ROR R0`

(12 marks)

(b) Write the equivalent VIP assembly language code that represents part of the C high-level language program instructions given in Figure Q1b. You may assume that the values of these C variables **R0**, **R1**, **R2** and **R3** are *unsigned numbers* located in the 12-bit VIP registers **R0**, **R1**, **R2** and **R3** respectively.

(10 marks)

```
R1=10;
R2=5;
R3=0;
for (R0=1; R0<=3; R0++)
{
    R1=R1+R0;
    R2=R1-R2;
    if (R1>=R1)
    {
        R3=R2+R3;
    }
}
```

Figure Q1b

(c) Describe clearly how your solution for part Q1(b) will need to be modified if the C variables **R0**, **R1**, **R2** and **R3** in Figure Q1b are now treated as *signed numbers*.

(3 marks)

2. (a) Describe clearly what is meant by passing parameter by reference and give two reasons why you may want to pass a parameter by reference.

(6 marks)

```
Main  MOV   SP,#0xFFF   ; Initialize stack pointer                    (I1)
      PSH   #15         ; Push number of elements to sum             (I2)
      PSH   #0x100      ; Push start address of array A              (I3)
      PSH   #0x200      ; Push start address of array B              (I4)
      PSH   #0x300      ; Push start address of array C              (I5)
      CALL  SumArray    ; Call subroutine to do C[i] = A[i]+B[i]     (I6)
      :     :           ; Instruction immediately after CALL         (I7)
      :     :
```

Figure Q2

(b) Figure Q2 shows a VIP assembly language program that sets up various parameters before calling the subroutine **SumArray**. The subroutine adds **15** consecutive elements of two arrays **A** and **B** that are stored in memory starting at addresses **0x100** and **0x200** respectively. The results of the addition **A[i]** + **B[i]** will be stored into a third array **C** at the corresponding element position **C[i]**. The array **C** starts at memory address **0x300**.

Write a VIP assembly language subroutine labeled **SumArray** to perform the required function stated above. Your subroutine must only use information from the parameters that have been pushed to the system stack, as shown in Figure Q2. Your solution must also ensure the subroutine is transparent.

(13 marks)

(c) You would like to determine the memory locations where the main program has been placed. With reference to Figure Q2, give the sequence of VIP assembly language instructions that you would place at (**I7**) onwards in order to compute the start address of the instruction at label **Main**. This address value should be placed into register **R0**. Your sequence of instructions must also remove the parameters that were pushed to the stack earlier.

(6 marks)

3. Figure Q3 shows a basic Sketch program used on the Arduino UNO R3 board to enable UART-based serial communication between the board and its interfacing computer.

```
void setup() {
    // 38400 baud rate
    // 1 start bit, 7 data bits
    // 1 even parity bit, 1 stop bit

    Serial.begin(38400,SERIAL_7E1);
}

// Initialize variable to ASCII character 'U'
int thisByte = 'U';

void loop() {

    Serial.write(thisByte);
    delay(1000);
}
```

Figure Q3

With reference to Figure Q3, answer the questions below.

(a) Describe the differences between polling, interrupt-driven and DMA data transfer techniques. State which technique is used in the UART communication.

(8 marks)

(b) Describe the differences between synchronous and asynchronous data transfer. State which data transfer method is used in the UART communication.

(6 marks)

(c) The ASCII upper-case character 'U' has hexadecimal value 68. What should be the value of its parity bit?

(2 marks)

(d) Determine the maximum data transfer rate in characters per second.

(4 marks)

Note: Question No. 3 continues on Page 5

(e) If the board is re-configured to send the same data packet with no parity, determine the minimum baud rate setting needed to transfer at least 3000 characters per second. The supported baud rates are 19200, 28800, 38400 and 57600. Justify your answer.

(5 marks)

4. (a) For each of the devices listed in Table Q4a, state whether its memory is volatile or non-volatile and categorize the device as either system or storage memory. Give your answers using the table format given in Table Q4a.

(6 marks)

Table Q4a

Device	Volatile/Non-volatile	System/Storage Memory
DDR SDRAM		
Magnetic Hard-Disk		
Register		

(b) Table Q4b is an observation table for several direct-mapped cache memory accesses. The address reference strings are provided as hexadecimal numbers and the order of memory access starts from **188** and ends at **27A**.

For each of these address reference strings,

- give the value of the binary bits in the Tag, Block and Word fields.
- state whether each cache memory access registers a hit or a miss, and if the cache content is replaced.

Assume the cache memory is initially empty. Provide your answers using the table format given in Table Q4b.

(12 marks)

Note: Question No. 4 continues on Page 6

Table Q4b

Direct-Mapped Cache Memory					
Address Reference String (Hex)	Main Memory Address			Cache Access (Hit/Miss?)	Cache Replace (Yes/No?)
	Tag (2 bits)	Block (4 bits)	Word (4 bits)		
188					
183					
172					
272					
27A					

(c) Figure Q4 shows the initial contents (in 12-bit hexadecimal format) of several registers in the VIP processor and a VIP program that is to be executed.

Figure Q4

After the program is executed, what are the hexadecimal contents of registers **R0**, **R2**, **R3** and **AR**?

(7 marks)

Appendix 1

VIP Instruction Set Summary Chart

VIP Instruction Encoding – Opcode Formats

11	10	9	8	7	6	5	4	3	2	1	0
0-7	Dual operand			d				s			
8	Short Move			d				n			
9-A	Unary/Control			op-code				operand = s, d or n			
B-F	JMP			2's complement -128 to +127 *relative*							

Group 1 – Dual-Operand Instructions (Opcode: 000 to 8FF)

Bits 8-11	Name	Bits 4-7	Bits 0-3	Operation	Flags
0	MOV	d	s	d ← s	NZ
1	AND	d	s	d ← d .AND. s	NZ
2	OR	d	s	d ← d .OR. s	NZ
3	EOR	d	s	d ← d .EOR. s	NZ
4	ADD	d	s	d ← d + s	VNZC
5	ADDC	d	s	d ← d + s + carry	VNZC
6	SUB	d	s	d ← d + (.NOT. s) + 1	VNZC
7	CMP	d	s	d + (.NOT. s) + 1	VNZC
8	MOVS	d	n	d ← n	

Group 2 – Unary and Control Instructions (Opcode: 900 to 9FF)

Bits 4-7	Name	Bit 0-3	Operation	Flags
0	INC	d	d ← d + 1	C
1	DEC	d	d ← d + 0xFFF	NZC
2	ROR	d	Rotate d right : msb ← lsb; and C ← lsb	NZC
3	ROL	d	Rotate d left : lsb ← msb; and C ← msb	NZC
4	RRC	d	Rotate d right including carry	NZC
5	RLC	d	Rotate d left including carry	NZC
6	RAR	d	Rotate d 'arithmetic' right preserving msb	NZC
7	PRSG	d	Left shift lsb from EOR (bits 11,5,3,0)	NZC
8	INV	d	d ← .NOT. d	NZ
9	NEG	d	d ← (.NOT. d) + 1	NZC
A	DADD	s	AR ← AR + s + carry (as 3 BCD digits)	ZC
B	UMUL	s	R1:R0 ← unsigned R0 times unsigned s	Z
C	TST	s	s + 0	NZ
D	EXEC	s	Execute s as an instruction	implied
E	BCSR	n	SR (bits 3-0) ← SR .AND. (.NOT. n)	explicit
F	BSSR	n	SR (bits 3-0) ← SR .OR. n	explicit

Group 3 – Unary and Control Instructions (Opcode: A00 to AFF)

Bits 4-7	Name	Bits 0-3	Operation	Flags
0	PSH	s	SP ← SP-1; (SP) ← s	
1	POP	d	d ← (SP); SP ← SP+1	explicit if d=SR
2	PSHM	3:2:1:0	Push R3:2:1:0 to stack, R3 first	
3	POPM	3:2:1:0	Pop R3:2:1:0 from stack, R3 last	
4	CALL	s	SP ← SP-1; (SP) ← Return Address PC ← Effective address	
5	RET	n	PC ← (SP) + n; SP ← SP+1	
6			See subgroup 3a	
7	RCN	n	Count for next rotate instruction. if n=0 use bits 3:2:1:0 of AR	
8	JDAR	±n	AR ← AR-1, if AR != 0, PC ← PC ± n	
9	JPE	±n	If parity of AR is even, PC ← PC ± n	
A	JPL	±n	If N = 0, PC ← PC ± n	
B	JVC	±n	If V = 0, PC ← PC ± n	
C	JGE	±n	If N = V, PC ← PC ± n	
D	JLT	±n	If N != V, PC ← PC ± n	
E	JGT	±n	If Z = 0 and N = V, PC ← PC ± n	
F	JLE	±n	If Z = 1 or N != V, PC ← PC ± n	

Group 1 – Jump Instructions (8-bit Range) (Opcode: B00 to FFF)

Bits 8-11	Name	n = Bits 0 to 7	Operation
B	JMP = BRA	-128 to +127	PC ← PC ± n
C	JEQ = JZ	-128 to +127	If Z=1, PC ← PC ± n
D	JNE = JNZ	-128 to +127	If Z=0, PC ← PC ± n
E	JHS = JC	-128 to +127	If C=1, PC ← PC ± n
F	JLO = JNC	-128 to +127	If C=0, PC ← PC ± n

Group 3a – Control Instructions (Opcode: A60 to A6F)

Bits 4-7	Name	Bits 0-3	Operation
6	RETI	0	SR ← (SP); SP ← SP+1; PC ← (SP); SP ← SP+1
6	SWI	1	SP ← SP-1; (SP) ← PC; SP ← SP-1; (SP) ← SR; PC ← (0x009)
6	WAIT	2	IE ← 1; Execution resumes after interrupt signal
6	HALT	3	Stop execution. Non-maskable interrupt or hardware reset to exit.
6	STOP	4	Stop execution. Reset to exit.
6	SYNC	8	Pulse SYNC output pin high for 1 clock cycle
6	NOP	9	No operation
6	LOCK	A	Block interrupts and bus sharing
6	UNLK	B	Allow interrupts and bus sharing
6	MSS	C to F	Memory Space Select override

Addressing Modes

Hex	Symbol	Location of Data	Availability
0	R0	Register R0	Both d and s
1	R1	Register R1	Both d and s
2	R2	Register R2	Both d and s
3	R3	Register R3	Both d and s
4	[R0]	Register R0 indirect	Both d and s
5	[R1]	Register R1 indirect	Both d and s
6	[R2+n]	Register R2 with offset indirect	Both d and s
7	[R3+n]	Register R3 with offset indirect	Both d and s
8	AR	Data is in Auxiliary Register	Both d and s
9	SR	Status Register	Both d and s
A	SP	Stack Pointer	Both d and s
B	PC	Program Counter	Both d and s
C	#n	Immediate, (or just n for CALL)	s only
D	[n]	Absolute (code space for CALL)	Both d and s
E	[SP+n]	SP with offset indirect	Both d and s
F	[PC+n]	PC with offset indirect (CALL is relative with PC+n)	Both d and s

Notation: d = destination; s = source

Description of bits in Status Register

SR	F	R	Description
11-8	*	*	Reserved
7-4	*	*	Defined but not described here
3	V	0	Set if 2's complement sign is incorrect
2	N	0	Is most significant bit of result
1	Z	0	1 if result is zero, otherwise 0
0	C	0	1 if carry out, otherwise 0

Notation: SR=Bits in register; F=Name of flag; R=Value after reset

END OF PAPER

CZ1006 COMPUTER ORGANISATION AND ARCHITECTURE

Please read the following instructions carefully:

1. **Please do not turn over the question paper until you are told to do so. Disciplinary action may be taken against you if you do so.**

2. You are not allowed to leave the examination hall unless accompanied by an invigilator. You may raise your hand if you need to communicate with the invigilator.

3. Please write your Matriculation Number on the front of the answer book.

4. Please indicate clearly in the answer book (at the appropriate place) if you are continuing the answer to a question elsewhere in the book.