ECED 3403 – Computer Architecture

Quiz 1

12 June 2024

The following quiz is closed book and notes. Internet-connected computing devices may not be used. You may use notes written or printed on both sides of a letter-sized paper (8.5" by 11" or 21.59 cm by 27.94 cm). Calculators are permitted. Answers are to be written directly on the quiz. State any assumptions made. Time allotted for quiz is 60 minutes. The mark associated with each question is written in parenthesis beside the question number.

1. (2) So far, we have not implemented instructions to change the flow of control in a program (BL through BRA). However, an XM23 program can jump to any location in instruction memory with some of the instructions we have implemented. How can this be done?

2. (2) The following code fragments add the number 10 to register R4:

Fragment 1:	Fragment 2:

`ADD   #8,R4`
`ADD   #2,R4`
	`MOVLZ      $10,R0`
	`ADD        R0,R4`

   Which fragment is preferable? Why?

3. (3) What is the difference between the BIT and the AND instructions? Give an example in XM23 assembler.

4. (3) What is a forward reference? Give an example of a forward reference in XM23 assembler. What takes place if a forward reference is detected?

5. (2) An application is performing signed integer arithmetic. If the CPU executes an addition and detects both overflow (V) and carry (C), what does the carry indicate?

6. (2) Is the following S-Record valid or invalid? Why?

```
S10B20000A40104CA24782497A
```

Table 1: XM-23p Instruction Set

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Mnemonic	Instruction
0	0	0	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	BL	Branch with Link
0	0	1	0	0	0	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	BEQ/BZ	Branch if equal or zero
0	0	1	0	0	1	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	BNE/BNZ	Branch if not equal or not zero
0	0	1	0	1	0	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	BC/BHS	Branch if carry/higher or same (unsigned)
0	0	1	0	1	1	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	BNC/BLO	Branch if no carry/lower (unsigned)
0	0	1	1	0	0	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	BN	Branch if negative
0	0	1	1	0	1	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	BGE	Branch if greater or equal (signed)
0	0	1	1	1	0	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	BLT	Branch if less (signed)
0	0	1	1	1	1	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	BRA	Branch Always
0	1	0	0	0	0	0	0	R/C	W/B	S/C	S/C	S/C	D	D	D	ADD	Add: DST ← DST + SRC/CON
0	1	0	0	0	0	0	1	R/C	W/B	S/C	S/C	S/C	D	D	D	ADDC	Add: DST ← DST + (SRC/CON + Carry)
0	1	0	0	0	0	1	0	R/C	W/B	S/C	S/C	S/C	D	D	D	SUB	Subtract: DST ← DST + (¬SRC/CON + 1)
0	1	0	0	0	0	1	1	R/C	W/B	S/C	S/C	S/C	D	D	D	SUBC	Subtract: DST ← DST + (¬SRC/CON + Carry)
0	1	0	0	0	1	0	0	R/C	W/B	S/C	S/C	S/C	D	D	D	DADD	Decimal add: DST ← DST + (SRC/CON + Carry)
0	1	0	0	0	1	0	1	R/C	W/B	S/C	S/C	S/C	D	D	D	CMP	Compare: DST – SRC/CON
0	1	0	0	0	1	1	0	R/C	W/B	S/C	S/C	S/C	D	D	D	XOR	Exclusive OR: DST ← DST ⊕ SRC/CON
0	1	0	0	0	1	1	1	R/C	W/B	S/C	S/C	S/C	D	D	D	AND	AND: DST ← DST & SRC/CON
0	1	0	0	1	0	0	0	R/C	W/B	S/C	S/C	S/C	D	D	D	OR	OR: DST ← DST | SRC/CON
0	1	0	0	1	0	0	1	R/C	W/B	S/C	S/C	S/C	D	D	D	BIT	Bit test: DST & (1 << SCR/CON)
0	1	0	0	1	0	1	0	R/C	W/B	S/C	S/C	S/C	D	D	D	BIC	Bit clear: DST ← DST & ~(1 << SRC/CON)
0	1	0	0	1	0	1	1	R/C	W/B	S/C	S/C	S/C	D	D	D	BIS	Bit set: DST ← DST | (1 << SRC/CON)
0	1	0	0	1	1	0	0	0	W/B	S	S	S	D	D	D	MOV	DST ← SRC
0	1	0	0	1	1	0	0	1	0	S	S	S	D	D	D	SWAP	Swap SRC and DST
0	1	0	0	1	1	0	1	0	W/B	0	0	0	D	D	D	SRA	Shift DDD right (1 bit) arithmetic
0	1	0	0	1	1	0	1	0	W/B	0	0	1	D	D	D	RRC	Rotate DDD right (1 bit) through carry
0	1	0	0	1	1	0	1	0	0	0	1	1	D	D	D	SWPB	Swap bytes in DDD
0	1	0	0	1	1	0	1	0	0	1	0	0	D	D	D	SXT	Sign-extend byte to word in DDD
0	1	0	0	1	1	0	1	1	0	0	0	0	PR	PR	PR	SETPRI	Set current priority
0	1	0	0	1	1	0	1	1	0	0	1	SA	SA	SA	SA	SVC	Control passes to address specified in vector[SA]
0	1	0	0	1	1	0	1	1	0	1	V	SLP	N	Z	C	SETCC	Set PSW bits (1 = set)
0	1	0	0	1	1	0	1	1	1	0	V	SLP	N	Z	C	CLRCC	Clear PSW bits (1 = clear)
0	1	0	1	0	0	C	C	C	C	T	T	T	F	F	F	CEX	Conditional execution
0	1	0	1	1	0	PRPO	DEC	INC	W/B	S	S	S	D	D	D	LD	DST ← mem[SRC plus addressing]
0	1	0	1	1	1	PRPO	DEC	INC	W/B	S	S	S	D	D	D	ST	mem[DST plus addressing] = SRC
0	1	1	0	0	B	B	B	B	B	B	B	B	D	D	D	MOVL	DST.Low byte ← BBBBBBBB; DST.High byte unchanged
0	1	1	0	1	B	B	B	B	B	B	B	B	D	D	D	MOVLZ	DST.Low byte ← BBBBBBBB; DST.High byte ← 00000000
0	1	1	1	0	B	B	B	B	B	B	B	B	D	D	D	MOVLS	DST.Low byte ← BBBBBBBB; DST.High byte ← 11111111
0	1	1	1	1	B	B	B	B	B	B	B	B	D	D	D	MOVH	DST.Low byte unchanged; DST.High byte ← BBBBBBBB
1	0	OFF	OFF	OFF	OFF	OFF	OFF	OFF	W/B	S	S	S	D	D	D	LDR	DST ← mem[SRC + sign-extended 7-bit offset]
1	1	OFF	OFF	OFF	OFF	OFF	OFF	OFF	W/B	S	S	S	D	D	D	STR	mem[DST + sign-extended 7-bit offset] ← SRC