Computer Organization & Assembly Language

Midterm Exam – 2019/11/14

Dept. of Engineering Science, National Cheng Kung University

1. Answer "True" or "False" to the following statements. (30%)
   (A) A single instruction in assembly language may map to one or more operations in machine codes.
   (B) The yield decreases as there are more and more defects on the wafer.
   (C) The performance of an antilock brake system can be evaluated in terms of the real-time constraint.
   (D) Benchmark refers to programs used to measure performance.
   (E) When representing a negative number, the MSB is used as the sign bit.
   (F) The "`shamt`" field is not used by the `add` instruction but is used by the `sw` instruction.
   (G) J-format (or J-type) instructions support conditional branching.
   (H) The data stored in `$t0~$t9` are never preserved during procedure calls.
   (I) The stack in MIPS grows from lower address to higher address.
   (J) The `nor` instruction can help on performing the logical NOT operation.

2. Suppose die area is 0.4 cm^2 and there are 4 defects per cm^2. Calculate the yield. Then calculate the yield if defects per area can be cut in half. Note that the answers should be in the form of xx.xx%. (8%)

$$Yield = \frac{1}{\left(1 + DefectsPerArea \bullet \frac{DieArea}{2}\right)^2}$$

3. Computer A has an overall CPI of 1.3 and can be run at a clock rate of 600MHz. Computer B has a CPI of 2.5 and can be run at a clock rate of 750MHz. We have a particular program we wish to run. When compiled for computer A, this program has exactly 100,000 instructions. How many instructions would the program need to have when compiled for Computer B, in order for the two computers to have exactly the same execution time for this program? (4%)

4. Assume for a given processor the CPI of arithmetic instructions is 1, the CPI of load/store instructions is 10, and the CPI of branch instructions is 3. Assume a program has the following instruction breakdowns: 500 million arithmetic instructions, 300 million load/store instructions, 100 million branch instructions.
   (A) Suppose that new, more powerful arithmetic instructions are added to the instruction set. On average, through the use of these more powerful arithmetic instructions, we can reduce the number of arithmetic instructions needed to execute a program by 25%, and the cost of increasing the clock cycle time by only 10%. Is this a good design choice? Why? (6%)
   (B) Suppose that we find a way to double the performance of arithmetic instructions. What is the overall speedup of our machine? What if we find a way to improve the performance of arithmetic instructions by 10 times? (4%)

5. What kind of instructions is supported by PC-relative addressing? Explain in detail how PC-relative addressing works. (8%)

6. Compile the following C program into MIPS instructions. Assume that the usage of registers is specified as (`f: $s0`). (12%)

```
int function_x (int* a, int h){
    int f;
    if(h >= 0)   f = a[10] + h;
    else         f = a[10] - h;

    return f;
}
```

7. Execute the following MIPS code fragments, showing the changes that occur in the register file and in memory. You only need to show the changes. (12%)

(A)

```
addi    $19, $0, 0x20
lw      $17, 0x04($19)
add     $20, $19, $16
sw      $20, 0x08($19)
```

Registers	BEFORE	AFTER
$16	0x10	×
$17	0x14	
$18	0x16	×
$19	0x28	
$20	0x1234	

Memory	BEFORE	AFTER
0x20	0x22	×
0x24	0x30	×
0x28	0x40	
0x2C	0x50	×
0x30	0x60	×

(B)

```
addi    $1, $0, 0x20
lw      $2, 0x04($3)
add     $0, $3, $1
bne     $0, $1, loop
```

Registers	BEFORE	AFTER
$0	0x00	×
$1	0x14	
$2	0x16	
$3	0x28	×
$4	0x1234	×

Memory	BEFORE	AFTER
0x20	0x10	×
0x24	0x30	×
0x28	0x40	×
0x2C	0x50	×
0x30	0x60	×

(C) Is the branch taken? (YES or NO)

8. Given a 32-bit bit pattern: 00100010 01010001 00101010 01010010 (16%)
   (A) What is the corresponding hexdecimal representation if it is an integer?
   (B) What is the corresponding string if it is an ASCII string?
   (C) What is the corresponding instruction if it is an MIPS instruction?
   (D) Assume that `$t0=$t1=$t2=$t3=`23_{ten} and `$s0=$s1=$s2=$s3=`51_{ten}. If the instruction decoded in (C) is then executed, which register is updated? What is the new value (in decimal) for this register?

References:

ASCII value	Character	ASCII value	Character	ASCII value	Character	ASCII value	Character	ASCII value	Character	ASCII value	Character
32	space	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	DEL

instruction	format	op	rs	rt	rd	shamt	funct	address
add	R	0	reg	reg	reg	0	32_{ten}	n.a.
sub	R	0	reg	reg	reg	0	34_{ten}	n.a.
addi	I	8_{ten}	reg	reg	n.a.	n.a.	n.a.	constant
lw	I	35_{ten}	reg	reg	n.a.	n.a.	n.a.	address
sw	I	43_{ten}	reg	reg	n.a.	n.a.	n.a.	address