TEST

(Version 1:4)

COMPSCI.210.F.T
Computer Systems

29th April 2002, 13:35 - 14:25pm

(TIME ALLOWED: 50 MINUTES)

DO NOT START, DO NOT OPEN SCRIPT!

UNTIL INSTRUCTED TO DO SO.

Please write your family name, given name and student ID at the top of every page. Answer all questions on the test paper in the spaces provided. The test is out of 100 (as a guide: allow approx 1 minute for every 2 marks). The test is worth 10% of your final grade.

No calculators are allowed!

There are three parts to the test. Part A (worth 30%) is on Data Representation Part B (worth 20%) is on UNIX and Part C (worth 50%) is on the alpha assembly language.

Print name clearly: _____

PART A: Number representation (worth 30% = 30 Marks)

A. 1

Convert the following decimal numbers first to binary and then to hexadecimal: *[4 marks]*

23_{10} 123_{10}

10111_2	1111011_2
17_{16}	$7B_{16}$

A. 2

Perform the following operations in 8-bit two's complement arithmetic showing explicitly the overflow and carry outcomes: *[6 marks]*

```
  0110 1100          1101 0011
+ 0011 0101        + 1011 0100
```

(0) 1010 0001 overflow!	(1) 1000 0111 no overflow!

A. 3

Convert the following unsigned hexadecimal numbers to octal: *[4 marks]*

225_{16} $1EC3_{16}$

001|000|100|101|	*0|001|111|011|000|011|*
1045_8	017303_8

A. 4

Convert the decimal -43.75 into binary floating point assuming the following 16 bit format: the sign bit is in MSB position. The next five (5) bits are allocated to storing the exponent which assumes an XS31 representation, i.e., offset binary *with* $k = 31$*, and the remaining ten (10) LSB's record the fraction, stored according to the IEEE normalisation convention, i.e.,* $1.M$*.* *[6 marks]*

Now this question had a problem which only about 15% or so of students recognised. In setting $k = 31$ *where the exponent is limited to 5 bits, means the floating point format supported here is asymmetric Not what was really intended.*

-43.75 *implies sign bit* $= 1$, $43.75_{10} = 101011.11_2 = 1.0101111 \times 2^5$

Thus the fractional part will be 0101111000*, the exponent is* $31 + 5 = 36 = 100100_2$ *clearly one bit too big for the allotted 5 bits. Presumably in the current format this is truncated to 5 bits: the result is:* 1 00100 0101111000

A. 5

What is the worst case error implicit in the floating point format described in the previous question? You may express this either as a ratio, or as a percentage. *[3 marks]*

The largest error may arise from round-off, a '1' in the in the fractional part rounding in to the 10th place. Since we assume the smallest fractional value is 1.0000000000 *then the most the error can be is* $\pm 2^{-11}/1.0000000000$. *Expressed as percentage this is* $2^{-11} \times \frac{100}{1}\% \approx .05\%$

A. 6

Take the 2's complement of the following 8-bit binary numbers: *[4 marks]*

`0010 1101` `1000 0111`

```
 1101 0010          0111 1000
       +1                 +1
 1101 0011          0111 1001
```

A. 7

Convert the 4 characters "COMP" into the corresponding ASCII encoded, null delimited, hexadecimal string (see Appendix A): *[3 marks]*

From Appendix A we have the letters C O M P with ASCII hex codes given in lower left of each box, thus COMP null delimited encodes as $434F4D5000$

hexadecimal. [7 marks]

the output will be: `bin  file4  src  unixbook`

b. `cd ../.. ; ls`

PART B: Subsystem components, UNIX, etc (worth 20% = 20 Marks)

terms:

`harry  joe  mydir` [4 marks]

B. 1

Following is a diagram that shows a file structure commencing at the root directory.

Print name clearly: _____

c. `cd .. ; ls`

```
harry     joe     mydir
```

d. `echo *`

```
bin     src     unixbook
```

B. 2

Assume that a file called `fruit` *contains a list:*

```
apples
oranges
peaches
bananas
lemons
peaches
apricots
grapefruit
lemons.
```

Give a shell command (not a procedure) that converts the contents of the file to a sorted list, excluding duplicates, and saves this in a new file called `fruit_list`.

[4 marks]

```
sort -u fruit > fruit_list
```

command `newvar = string`. Give a command that will display

Print name clearly: _____

B. 3

Give a grep *command that will output, from the file* `fruit_list` *(see previous question)*

[4 marks]

a. all words in the list containing the letter `a`

`grep a fruit_list` *or* `grep "a" fruit_list`

b. all words containing at least two `a`*'s*

```
grep "a.*a" fruit_list
```

B. 4

Write down UNIX shell commands to achieve each of the following tasks:

[6 marks]

a. To combine two text files `Chapt1` *and* `Chapt2` *into a new single file called* `book`*:*

```
cat Chapt1 Chapt2 > book
```

b. To initiate a remote terminal session with a UNIX host called `m3r.tcs.auckland.ac.nz`*:*

```
telnet m3r.tcs.auckland.ac.nz
```

c. To change the current working directory to its parent directory:

```
cd ..
```

a screen-full at a time.

Part C (worth 50%)

C. 1

For each of the following, answer with a simple Yes *or* No. *[3 marks per correct answer]*

1. `beq $a0, end;` *implies, if a0 is equal to 1, branch to the label* `end`.

No. You will branch to label *end* only if $a0 equal 0

2. The instructions `addq $T0, $T1` *and* `addq $T0, $T1, $T1` *do the same thing.*

No. The first does $T0 = T0 + T1$, the second does $T1 = T0 + T1$

3. On the alpha, the 64 integer registers are each composed of 32 bits.

No. On the alpha, the 32 integer registers are each composed of 64 bits

4. Both `br` *and* `bsr` *instructions modify the $RA register (return address register).*

No. The $RA register holds the Program counter register content when the *bsr* instruction is called. The *br* instruction does not change the $RA register.

5. If I do not answer this question correctly, I'll get 3 marks.

To obtain 3 marks, you need to answer this question correctly.
Accepted answers were No or False.

C. 2

Suppose we have the following data in memory:

```
Location:     Contents:
0x1000000     0x123456789abcdef0
0x1000008     0x9988776655443322
```

Fill out the table below to show, in hexadecimal, the given contents of memory (assume memory is byte addressable): *[10 marks]*

memory address	*contents*	*memory address*	*contents*
0x1000000	*f0*	*0x1000008*	*22*
0x1000001	*de*	*0x1000009*	*33*
0x1000002	*bc*	*0x100000a*	*44*
0x1000003	*9a*	*0x100000b*	*55*
0x1000004	*78*	*0x100000c*	*66*
0x1000005	*56*	*0x100000d*	*77*
0x1000006	*34*	*0x100000e*	*88*
0x1000007	*12*	*0x100000f*	*99*

Assuming the initial memory content from the previous question, fill out the table below to show, in hexadecimal, the amended contents of memory, as well as the register contents, after the execution of the instructions listed below: *[25 marks]*

```
ldiq    $T0, 0x1000000; T0 = 0x1000000
ldw     $T1, 2($T0); T1 = 0xfffffffffff9abc
ldbu    $T2, 6($T0); T2 = 0x34 or T2 = 0x0000000000000034
sll     $T1, 56, $T3; T3 = 0xbc00000000000000 (shift 56 bits to the left)
sra     $T3, 8, $T4; T4 = 0xffbc000000000000 (shift 8 bits to the right.Mind the MSB.)
addq    $T3, $T4, $T5; T5 = 0xbbbc000000000000 (No overflow)
xor     $T4, $T5, $T6; T6 = 0x4400000000000000 (Exclusive or)
cmpeq   $T5, $T6, $T7; T7 = 0x0 (T5 and T6 are not equal)
cmovne  $T7, -5, $T7; T7 = 0x0 (conditional move only if T7 is not equal to 0)
stl     $T1, ($T0); (store 4 bytes at starting address 0x1000000)
stw     $T7, 0x8($T0); (store 2 bytes at starting address 0x1000008)
stw     $T2, 5($T0); Storing address not aligned to word size
```

memory address	*contents*	*memory address*	*contents*	*register*	*register contents*
0x1000000	*bc*	*0x1000008*	*00*	*T0*	*0x1000000*
0x1000001	*9a*	*0x1000009*	*00*	*T1*	*0xfffffffffff9abc*
0x1000002	*ff*	*0x100000a*	*44*	*T2*	*0x0000000000000034*
0x1000003	*ff*	*0x100000b*	*55*	*T3*	*0xbc00000000000000*
0x1000004	*78*	*0x100000c*	*66*	*T4*	*0xffbc000000000000*
0x1000005	*56*	*0x100000d*	*77*	*T5*	*0xbbbc000000000000*
0x1000006	*34*	*0x100000e*	*88*	*T6*	*0x4400000000000000*
0x1000007	*12*	*0x100000f*	*99*	*T7*	*0x0*

Print name clearly: _____

Rough working area (will not be marked).

Print name clearly: _____

Rough working area (will not be marked).

Appendix A

b7 b6 b5 / BITS / b4 b3 b2 b1	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
	CONTROL		SYMBOLS NUMBERS		UPPER CASE		LOWER CASE	
0 0 0 0	0 NUL 0	16 DLE 20	32 SP 40	48 0 60	64 @ 100	80 P 120	96 ‘ 140	112 p 160
0 0 0 1	1 SOH 1	17 DC1 21	33 ! 41	49 1 61	65 A 101	81 Q 121	97 a 141	113 q 161
0 0 1 0	2 STX 2	18 DC2 22	34 ” 42	50 2 62	66 B 102	82 R 122	98 b 142	114 r 162
0 0 1 1	3 ETX 3	19 DC3 23	35 # 43	51 3 63	67 C 103	83 S 123	99 c 143	115 s 163
0 1 0 0	4 EOT 4	20 DC4 24	36 $ 44	52 4 64	68 D 104	84 T 124	100 d 144	116 t 164
0 1 0 1	5 ENQ 5	21 NAK 25	37 % 45	53 5 65	69 E 105	85 U 125	101 e 145	117 u 165
0 1 1 0	6 ACK 6	22 SYN 26	38 & 46	54 6 66	70 F 106	86 V 126	102 f 146	118 v 166
0 1 1 1	7 BEL 7	23 ETB 27	39 ’ 47	55 7 67	71 G 107	87 W 127	103 g 147	119 w 167
1 0 0 0	8 BS 10	24 CAN 30	40 (50	56 8 70	72 H 110	88 X 130	104 h 150	120 x 170
1 0 0 1	9 HT 11	25 EM 31	41) 51	57 9 71	73 I 111	89 Y 131	105 i 151	121 y 171
1 0 1 0	10 LF 12	26 SUB 32	42 * 52	58 : 72	74 J 112	90 Z 132	106 j 152	122 z 172
1 0 1 1	11 VT 13	27 ESC 33	43 + 53	59 ; 73	75 K 113	91 [133	107 k 153	123 { 173
1 1 0 0	12 FF 14	28 FS 34	44 , 54	60 < 74	76 L 114	92 \ 134	108 l 154	124 | 174
1 1 0 1	13 CR 15	29 GS 35	45 − 55	61 = 75	77 M 115	93] 135	109 m 155	125 } 175
1 1 1 0	14 SO 16	30 RS 36	46 . 56	62 > 76	78 N 116	94 ^ 136	110 n 156	126 ~ 176
1 1 1 1	15 SI 17	31 US 37	47 / 57	63 ? 77	79 O 117	95 _ 137	111 o 157	127 DEL 177

LEGEND: dec / CHAR / hex oct

Figure 1: American Standard Code for Information Interchange (ASCII)