CMPE12 — Computing systems and assembly language, Fall 2006
Midterm 2

Tuesday November 28th, 2006

No books, notes or calculator.
Simple computations are expected to be carried out by hand.
Justify all your answers — no credit will be given for a correct answer if work is not shown
Use the back of pages as scratch paper, if necessary.

<table>
<thead>
<tr>
<th>Exercise</th>
<th>Score</th>
<th>Out of</th>
</tr>
</thead>
<tbody>
<tr>
<td>M2.1</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>M2.2</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>M2.3</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>M2.4</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>M2.5</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>TOT</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
M2.1 Below is an LC3 program.

```
.orig    x3000
LD       R1, x
LD       R2, y
AND      R3, R3, #0
JSR      TWEETY
TOM      BRzp   JERRY
         ADD     R1, R1, #1
         JSR      TWEETY
JERRY    BRnz    SYLVESTER
         ADD     R1, R1, #-1
SYLVESTER ADD     R3, R3, #1
         JSR      TWEETY
         BRnp     TOM
         ST       R3, z
         halt
x .fill   x0000
y .fill   x0008
z .fill   x0000
.blkw     4
TWEETY   NOT     R0, R2
         ADD     R0, R0, #1
         ADD     R0, R1, R0
         RET
.end
```

- (5 points) What standard flow-control structure(s) does this program implement?

- (10 points) Write a high-level language fragment (C is preferred) equivalent to this program, using the appropriate control structures and making the code as concise and clean as possible.
M2.2 (10 points) Below is a snapshot of the LC3 simulator on which I have just finished running the program on page 2 in which I have changed the initial values of x, and y. I have accidentally erased the initial value of y from the figure. What was it?

```
R0 x0000 0  R4 x0000 0  PC x300D 12301
R1 x001C 28  R5 x0000 0  IR x3603 13827
R2 x001C 28  R6 x0000 0  PSR x0002 -32766
R3 x000C 12  R7 x300B 12299  CC Z

x3000 0110000000001100 x20D  LD R1, x
x3001 0110000000001100 x24D  LD R2, y
x3002 0101101011000000 x56E0  AND R3, R3, #0
x3003 0101000000000001 x4811  JSR TWEETY
x3004 0000001100000000 x0620  TOM BR2P JERRY
x3005 0001001001110000 x1261  ADD R1, R1, #1
x3006 0100100000001110 x480E  JSR TWEETY
x3007 0000100000000000 x0C01  JERRY BNNZ SYLVESTER
x3008 0001000000000001 x127F  ADD R1, R1, #-1
x3009 0001011011110000 x16E1  SYLVESTER ADD R3, R3, #1
x300A 0100100000000000 x480A  JSR TWEETY
x300B 0000101111111000 x0BF8  BNNP TOM
x300C 0011101110000011 x3603  ST R3, z
x300D 111100000010101 xF025  TRAP HALT
x300E 0000000000010000 x0010  x  NOP
x300F 0000000000000000  y  NOP
x3010 0000000000000000  z  NOP
x3011 0000000000000000  x  NOP
x3012 0000000000000000  x  NOP
x3013 0000000000000000  x  NOP
x3014 0000000000000000  x  NOP
x3015 1001000011111111 x90BF  TWEETY NOT R0, R2
x3016 0010000000010000 x1021  ADD R0, R0, #1
x3017 0010000000100000 x1040  ADD R0, R1, R0
x3018 1100000111100000 xC1C0  RET
x3019 0000000000000000  x  NOP
x301A 0000000000000000  x  NOP
x301B 0000000000000000  x  NOP
x301C 0000000000000000  x  NOP
x301D 0000000000000000  x  NOP
x301E 0000000000000000  x  NOP
x301F 0000000000000000  x  NOP
```

M2.3 (14 points) Fill out the *symbol table* for the program on page 2 with the entries defined in this code fragment. Also indicate what are the names of the two columns of the symbol table.

<table>
<thead>
<tr>
<th>(what is in this col?)</th>
<th>(what is in this col?)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
M2.4 (16 points) The picture below describes the implementation of memory-mapped I/O in the LC3 architecture. Fill out all the shaded boxes with the corresponding block’s missing name.
M2.5 This is an HC11 program, with instructions’ descriptions on the side.

/************************************************************
 * midterm program for HC11
 * andre di blas, november 2006
 ************************************************************/

#include <v2_18g3.asm>

.sect .data

var1: .byte 0x12
var2: .byte 0x34
var3: .word 0x5678
var4: .byte 0x9A

/************************************************************/
.sect .text

main:
  ldaa var1 ; load accumulator
  inca ; increment
  ldab var1 ; load accumulator
decb ; decrement
  ldx #var1 ; load index register X
  xgdx ; exchange double accumul. and index register x
  jsr CONSOLEINT ; display integer in acc. d on the console
  jsr OUTCRLF ; send newline to console
  addd #1 ; add double accumulator
  xgdx ; exchange double accumul. and index register x
  std 0, X ; store double accumulator
  jsr CONSOLEINT ; display integer in acc. d on the console
  jsr OUTCRLF ; send newline to console
  ldd var3 ; load double accumulator
  jsr CONSOLEINT ; display integer in acc. d on the console
  jsr OUTCRLF ; send newline to console
  ldd #var4 ; load double accumulator
  jsr CONSOLEINT ; display integer in acc. d on the console
  jsr OUTCRLF ; send newline to console
  jsr GETCHAR ; wait before terminating the program

/************************************************************/

(continued on next page)
M2.5 (cont.) Knowing that the assembler is not using memory locations below 0x1000, answer the following questions:

- (5 points) What is the addressing mode used by this instruction?
  ldaa var1

- (5 points) What is the addressing mode used by this instruction?
  inca

- (5 points) What is the addressing mode used by this instruction?
  ldx   #var1

- (5 points) What is the addressing mode used by this instruction?
  jsr   CONSOLEINT

- (5 points) What is the addressing mode used by this instruction?
  addd  #1

- (5 points) What is the addressing mode used by this instruction?
  std  0, X

Knowing that the first numbers that is written to the console is 16384,

- (5 points) What is the second number written to the console?

- (5 points) What is the third number written to the console?

- (5 points) What is the fourth number written to the console?
LC3 instruction set — this page can be detached but it must be returned with the test

| Format of the entire LC3 instruction set. NOTE: * indicates instructions that modify condition codes |

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD*</td>
<td>Add DR to SR1 with carry, store result in SR1</td>
</tr>
<tr>
<td>ADD+</td>
<td>Add DR to SR1 with overflow, store result in SR1</td>
</tr>
<tr>
<td>AND*</td>
<td>Logical AND of DR and SR1, store result in SR1</td>
</tr>
<tr>
<td>AND+</td>
<td>Logical AND of DR and SR1 with overflow, store result in SR1</td>
</tr>
<tr>
<td>BR</td>
<td>Branch to program offset PCoffset9</td>
</tr>
<tr>
<td>JMP</td>
<td>Jump to program offset BaseR</td>
</tr>
<tr>
<td>JSR</td>
<td>Call subroutine and return to program offset PCoffset11</td>
</tr>
<tr>
<td>JSRR</td>
<td>Return from subroutine</td>
</tr>
<tr>
<td>LD</td>
<td>Load data register with program offset PCoffset9</td>
</tr>
<tr>
<td>LDI</td>
<td>Load immediate data register with value 10</td>
</tr>
<tr>
<td>LDR*</td>
<td>Load data register with offset from BaseR</td>
</tr>
<tr>
<td>LEA*</td>
<td>Load effective address register with value 1110</td>
</tr>
<tr>
<td>NOT*</td>
<td>Negate data register and store result in data register</td>
</tr>
<tr>
<td>RET</td>
<td>Return from interrupt</td>
</tr>
<tr>
<td>RTI</td>
<td>Return from trap and clear trap flag</td>
</tr>
<tr>
<td>ST</td>
<td>Store data register with program offset PCoffset9</td>
</tr>
<tr>
<td>STI</td>
<td>Store immediate data register with value 101</td>
</tr>
<tr>
<td>STR</td>
<td>Store data register with offset from BaseR</td>
</tr>
<tr>
<td>TRAP</td>
<td>Trigger a trap with trap event 1111</td>
</tr>
<tr>
<td>reserved</td>
<td>Reserved instruction</td>
</tr>
</tbody>
</table>

TRAP Service Routines

<table>
<thead>
<tr>
<th>Trap Vector</th>
<th>Assembly Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x20</td>
<td>GETC</td>
<td>Read a single character from the keyboard into R0[7:0], no echo</td>
</tr>
<tr>
<td>x21</td>
<td>OUT</td>
<td>Write the character in R0[7:0] to the display</td>
</tr>
<tr>
<td>x22</td>
<td>PUTS</td>
<td>Write to display the 0-terminated string starting at address R0</td>
</tr>
<tr>
<td>x23</td>
<td>IN</td>
<td>Write a prompt to display and then read a char from keyb, returned in R0[7:0], with echo</td>
</tr>
<tr>
<td>x24</td>
<td>PUTSP</td>
<td>Write to display the packed (2 characters/word) 0-terminated string starting at address R0</td>
</tr>
<tr>
<td>x25</td>
<td>HALT</td>
<td>Halts execution and prints a message to the display</td>
</tr>
</tbody>
</table>

Device Register Assignments

<table>
<thead>
<tr>
<th>Device Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xFE00 KSR</td>
<td>Keyboard status register: bit 15 set means new character waiting</td>
</tr>
<tr>
<td>xFE02 KBBR</td>
<td>Keyboard buffer register: bits [7:0] contain the new character</td>
</tr>
<tr>
<td>xFE04 DSR</td>
<td>Display status register: bit 15 set means display ready for output</td>
</tr>
<tr>
<td>xFE06 DDR</td>
<td>Display data register: bits [7:0] are output on the screen</td>
</tr>
<tr>
<td>xFFFE MCR</td>
<td>Machine control register: bit 15 is the clock enable bit</td>
</tr>
</tbody>
</table>