MASSACHUSETTS INSTITUTE OF TECHNOLOGY
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

6.004 Computation Structures
Fall 2003

Quiz #3: October 31, 2003

Name	*Athena Username*	*Score*
DAN PORTS	dr kp	4+5+15=24

☐ WF 10, 34-304 Morris	☐ WF 12, 34-302 Fitzpatrick	☐ WF 1, 34-304 Asanovic
☐ WF 11, 34-303 Morris	☒ WF 12, 34-303 Mazzola Paluska	☐ WF 2, 34-303 Asanovic
☐ WF 11, 36-372 Fitzpatrick	☐ WF 1, 34-303 Mazzola Paluska	

Problem 1 (5 points): **Beta Testing**

(A) In a Beta assembly-language program, does writing

ADDC(R0, 17, R1)

rather than

ADDC(R0, 3*4+5, R1)

make the resulting binary program smaller? Run faster?

Smaller program? YES: ____; NO: X

Faster program? YES: ____; NO: X

(B) Write a **single** assembly-language instruction that copies the TOPMOST word on the stack into R3 (without changing SP):

-1 ADDC(SP, 0, R3)
LD(SP, -4, R3)

(write Beta instruction)

(C) Is it true that, on the Beta, every SUBC instruction can be replaced by an equivalent ADDC instruction? SUBC(Ra, -32768, Rb) can't.

SUBC can be replaced? YES: ____; NO: X

(D) (2 points) In a Beta program, a BR instruction stored at address 0x80000 branches to an instruction at address 0x80000+x. What's the maximum (positive) value that x might have? Give your answer in HEX.

4 + 4·max(16 bit two's comp literal)
= 4 + 4(2^15 - 1)
= 2^17 = 2·2^16 = 2·16^4 = 0x20000

Maximum x, in HEX: 0x20000

Problem 2. (5 points, 1 point each): **Computable Functions**

Each of the following describes a machine which performs a specific function. In each case, identify whether the function can be implemented using an FSM, using a universal Turing Machine, or not at all. **Circle exactly one answer** for each function (answer FSM if the function can be implemented via either a Turing Machine or FSM).

A. A processor which, when connected to appropriate external memory, executes the Beta instruction set. Your answer should describe just the processor (including its registers and state), excluding other portions of the system like the memory.

β has finite state when main memory is excluded.

Circle one: can implement as **FSM** **TM** **uncomputable**

B. A device which takes as input the digits of a binary integer from left to right. The device has a single output which is 1 if and only if the number entered thus far is divisible by N, where N is a constant designed into the device.

N states

Circle one: can implement as **FSM** **TM** **uncomputable**

C. A device that takes a sequence of binary digits, one each millisecond clock period, and outputs 1 if and only if the sequence entered thus far contains more 1s than 0s.

Requires unbounded state → TM

Circle one: can implement as **FSM** **TM** **uncomputable**

D. Investment advisor: a device that takes as input an integer n between 0 and 200, and outputs the closing price of IBM stock during the n^{th} trading day of 2004 (to the nearest whole dollar).

Lookup table

Circle one: can implement as **FSM** **TM** **uncomputable**

E. A machine which takes a binary input i and outputs 1 if executing the i^{th} Turing machine on a blank (all zero) input tape halts, and outputs zero otherwise.

Circle one: can implement as **FSM** **TM** **uncomputable**

Suppose it's computable. $\forall i \; \forall j$ there is a Turing machine k that puts j on the tape then runs Turing machine i. Apply this machine to F_{IC}. Then it outputs 1 if $T_i(j)$ halts and zero otherwise. This solves the halting problem. So it's uncomputable.

Problem 3. (15 points): **Software Reverse Engineering**

You are given the following incomplete listing of a C program and its translation to Beta assembly code:

```
int f(int a, int b)
 { int c = a*b;
   if (a < b)
     return ????;
   else return c;
 }
```

```
f:  PUSH(LP)
    PUSH(BP)
    MOVE(SP, BP)
    ALLOCATE(1)
    PUSH(R4)
    PUSH(R5)

    LD(BP, -12, R4)
    LD(BP, -16, R5)
    MUL(R4, R5, R0)
    ST(R0, 0, BP)

    CMPLT(R4, R5, R0)
    BF(R0, L1)

    SUBC(R5, 1, R0)
    PUSH(R0)
    ADDC(R4, 1, R0)
    PUSH(R0)
    BR(f, LP)
xx: DEALLOCATE(2)
    BR(L2)

L1: LD(BP, 0, R0)
L2: POP(R5)
    POP(R4)
    MOVE(BP, SP)
    POP(BP)
    POP(LP)
    JMP(LP)
```

(A) (1 point) What is the total number of machine instructions comprised by the assembly program shown above? Give your answer **in decimal.**

Total program size in INSTRUCTIONS: 36

(B) (1 point) Within the body of f, does the address held in BP represent the location of some variable in the C program? If so, give the variable; else write "NONE".

Variable name, or NONE: C

(C) (2 points) What is the missing C source corresponding to the ???? in the above listing?

f(a+1, b-1)

(give C source fragment)

(D) (2 points) Give a HEX value for the instruction bearing the tag 'L1' in the above listing.

HEX translation of instruction at L1: 0x601B0000

In a test run, f(2, 5) is called from an external procedure and its execution is interrupted just prior to the execution of the instruction tagged 'xx'. The contents of a range of 4-byte memory locations corresponding to a portion of the stack are shown to the left. The contents of **BP** and **SP** are addresses of the indicated locations. Recall that the Beta uses byte addressing; word locations shown have addresses differing by 4.

NB: **All addresses and data values are shown IN HEX.** Please be sure to specify all of your answers **IN HEX.**

b	5	
a	2	
older LP	B4	
older BP	0	
C=>	???	110
R4	37	114
R5	23	118
b	4	11C
a	3	120
old LP	60	124
old BP	110	128
BP->	C	12C
R4	2	
R5	5	
next b	3	
next a	4	
SP ->		

(E) (1 point) What were the arguments to the current call to f?

Arguments (HEX): a=**0x** 3 ; b=**0x** 4 √+1

(F) (1 point) What value is in R0?

R0 = f(3,3) = 12 = 0xC

Contents of R0 (HEX): **0x** C √+1

(G) (1 point) What is the hex address of the instruction tagged xx?

Address of xx (HEX): **0x** 60 √+1

(H) (1 point) What is the missing value marked ??? on the stack? Give a hex number.

??? (HEX): **0x** A √+1

(I) (2 points) What is the hex address of the BR instruction that called f originally?

B4 is next instr, so B4 - 4 = B0

Address of original call: **0x** B0 √+2

(J) (1 point) What were the contents of R4 at the time of the original call? Answer NONE if you can't determine it.

Original contents of R4, in HEX or NONE: **0x** 37 √+1

(K) (2 points) What are the contents of BP?

Contents of BP (HEX): **0x** 12C √+2

END OF QUIZ!

+9