Daniel Leeds, 6.004 R15, April 12, 2006; Quiz #3 Review

**Fine print:**
Quiz is closed-book, no calculators; covers up to L14 (Stacks and Procedures)/R15 (this recitation)

**Practice, practice, practice:**
Follow “Previous terms” link from http://6004.csail.mit.edu, pick a semester (the more recent, the better), click on the “Announcements” page for the semester, and find the PDF for Quiz 3 solutions. Don’t read the answers until you first figure them out for yourself!

**Another perspective on the material – Margaret Chong’s Handbook:**
Follow “Handouts” link from http://6004.csail.mit.edu, click on handbook link near the bottom of the page.

**Handouts**
In past years, we have given students the “Summary of Instructions Format” sheet (available from the handouts web page) as a reference during the exam. While studying for the exam, you also might want to look at “Beta Documentation” – this won’t be available on test day.

**Good topics to know:**

*Models of Computation*

Turing Machines (TMs) – more powerful than FSMs
- Implementation: FSM attached to infinite tape
- Parenthesis checker – requiring arbitrarily many states
- Universal TMs capable of performing the computation performed by any TM
- Can compute all “computable” functions
- Uncomputable functions – for example, will TM $k$ ever halt on tape $j$?
  (Note “will TM $k$ halt on tape $j$ in fewer than $m$ steps” is computable)

*Programmable Machines and Machine Language*

Memory
- stores both data and coded instructions in words of $W$ bits (we tend to use 32 bits (= 4 bytes) per word)
- Program Counter (PC) specifies address of next instruction to be executed

Binary layout of the two Beta instruction formats:

<table>
<thead>
<tr>
<th>OPCODE</th>
<th>$r_c$</th>
<th>$r_a$</th>
<th>$r_b$</th>
<th>unused</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>OPCODE</th>
<th>$r_c$</th>
<th>$r_a$</th>
<th>16-bit signed constant</th>
</tr>
</thead>
</table>

Instructions
- move data between memory and registers
- operate on register data and store results in registers
- change program counter (for loops, procedure calls, conditional statements)

Sample Beta ops:

BEQ(R1, br_addr, R2) \hspace{2cm} R2 = PC+4 \hspace{2cm} **Machine Language Format**
branch relative to R2 -> \hspace{2cm} PC = R2 + 4*[(br_addr – R2)/4], if R1==0
\hspace{2cm} PC = R2, if R1!=0

OPCODE $r_c$ $r_a$ 16-bit literal \hspace{2cm} **Binary Encoding Format**
011101 00010 00001 0000000000001010 \hspace{2cm} (PC = PC+4+4*literal, if R1==0)
LD(R1,c,R2) \hspace{2cm} R2 = Mem[R1+c] (load value at address R1+c into R2)
Stacks and Procedures

Special registers:  
- **BP**: Base pointer is a reference point in the most recent activation record in the stack
- **LP**: Linkage pointer specifies return address for JMP at end of procedure call
- **SP**: Stack pointer points to top of the stack

Operations:  
- PUSH, POP
- ALLOCATE, DEALLOCATE (moves SP without read or write to memory)

Typical procedure call:

```
  . = 0x000000708
    PUSH(R2)      | CALL SEQUENCE
    BR(fact,LP)    |

  . = 0x00000B04  | later in memory
    PUSH(LP)      |     B38 LP
    PUSH(BP)      |     1058 BP
    MOVE(SP,BP)   |     2 R1
    PUSH(R1)      |     1 input
    LD(BP,-12,R1) | reads input left by
      | caller 3 address slots
      | above BP
    | details of fact omitted

  . = 0x00000B44  | at end of fact
    POP(R1)       | 2 input
    MOVE(BP,SP)   |
    POP(BP)       | B38 LP
    POP(LP)       |
    JMP(LP)       | 1068 BP
    BP-> 1 R1
    SP-> EDED
```

---

Higher points in computability theory (unlikely to be tested, but people asked in class)

Showing something is not computable often employs proof by contradiction - assume the function is computable, use it as part of a more complex Turing machine to construct a Turing machine that does something clearly impossible (recall the TM, $T_N$, that “halts when it doesn’t halt and doesn’t halt when it halts”). There are various non-computable functions, but you usually see them in theory papers, rather than computer consumer magazines.
Quiz #3 Spring 2003

Problem 1:
Identify whether the following behavior can be implemented using an FSM, a universal Turing Machine, or none at all. (Circle FSM if it can be implemented via either a Turing Machine or FSM).

A. A device that takes a stream of parentheses and outputs 1 if the input thus far represents a well-formed parenthesis string with no nesting (no (…) within (…)). It outputs a zero on mismatched or nested parens. FSM TM uncomputable

D. A machine that takes two binary inputs i and j and halts if and only if executing the i\textsuperscript{th} TM on tape j also halts. FSM TM uncomputable

Quiz #3 Spring 2004

Problem 3. (15 points): Digging into Beta code

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

```
int f(int a, int b)
{
    if (a < b)
        return f(a+a, b+1);
    else return ????
}
```

Note: while working this problem, you may wish to refer to the reference information (instruction set summary) attached to this quiz.

Give the HEX value of the instruction labeled ‘xx:’ in the program above.

(C) What is the missing C expression corresponding to the ‘???’ in the above C program?

(D) What would be the effect of removing the instruction MOVE(BP,SP)?

Procedure would work fine
Procedure would compute right value, but not restore registers correctly
Procedure would no longer compute \( f(a,b) \) properly
The call f(2, 5) is made via the instruction **BR (f, LP)** from an external main program and its execution is interrupted just prior to an execution (not necessarily the first) of the **BEQ** instruction labeled **xx**. The contents of a region of memory are shown to the right.

**NB:** All addresses and data values are shown in hex. The contents of **BP** are **0x128**.

<table>
<thead>
<tr>
<th>Address (HEX)</th>
<th>Contents (HEX)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>5</td>
</tr>
<tr>
<td>104</td>
<td>2</td>
</tr>
<tr>
<td>108</td>
<td>A8</td>
</tr>
<tr>
<td>10C</td>
<td>0</td>
</tr>
<tr>
<td>110</td>
<td>0</td>
</tr>
<tr>
<td>114</td>
<td>6004</td>
</tr>
<tr>
<td>118</td>
<td>6</td>
</tr>
<tr>
<td>11C</td>
<td>4</td>
</tr>
<tr>
<td>120</td>
<td>54</td>
</tr>
<tr>
<td>124</td>
<td>110</td>
</tr>
<tr>
<td><strong>BP ?</strong></td>
<td><strong>128</strong></td>
</tr>
<tr>
<td>12C</td>
<td>1</td>
</tr>
</tbody>
</table>

(E) (2 points) What are the arguments to the current (most recent) call to f?

Current arguments, a=_____; b=___

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

Contents of SP (HEX): 0x____

(G) (2 points) What is the address of the **BR (f, LP)** instruction that made the original call to f (2, 5)?

Address of BR making original call: 0x____

(H) (2 points) What value was in R2 at the time of the original call?