Please enter your name and Athena login name in the spaces above. Enter your answers in the spaces provided after each question. You can use the extra white space and the backs of the pages for scratch work.

**Problem 1. (5 points) Potpourri**

(A) (1 point) A complex combinational circuit is constructed entirely from 2-input NAND gates having a propagation delay of 1 ns. If this circuit is pipelined for maximal throughput by adding (non-ideal) registers whose setup time and propagation delay are each 1 ns, what is the throughput of the resulting pipeline? Enter a number, a formula, or “CAN’T TELL”.

\[
\text{Throughput (ns}^{-1}) = \frac{1}{3}
\]

(B) (1 point) Give the best achievable asymptotic throughput for a pipelined multiplier capable of multiplying two N-bit operands. Enter a number, a formula, or “CAN’T TELL”.

Asymptotic throughput: \( \Theta(\frac{1}{1}) \)

(C) (1 point) A “Thingee” is a clocked device built out of 3 interconnected components, each of which is known to be a 4-state FSM. What bound, if any, can you put on the number of states of a Thingee?

Max # of states, or “Can’t Tell”: \( 4^3 = 64 \)

(D) (2 points) A student tries to optimize his Beta assembly program by replacing a line containing

\[\text{ADDC(R0, 3*4+5, R1)}\]

by

\[\text{ADDC(R0, 17, R1)}\]

Is the resulting binary program smaller? Does it run faster?

(circle one) Binary program is SMALLER? yes ... no

(circle one) FASTER? yes ... no
Problem 2. (6 Points) Picking Locks, 6.004 style

Perfectly Perplexing Padlocks makes an entry-level electronic lock, the P3b, built from an FSM with two bits of state. The P3b has two buttons ("0" and "1") that when pressed cause the FSM controlling the lock to advance to a new state. In addition to advancing the FSM, each button press is encoded on the B signal (B=0 for button "0", B=1 for button "1"). The padlock unlocks when the FSM sets the UNLOCK output signal to 1, which it does whenever—and only whenever—the last 3 button presses correspond to the 3-digit combination. The combination is unique, and will open the lock independently of the starting state. Unfortunately the design notes for the P3b are incomplete.

(A) (1 Point) What is the 3-bit combination for the lock?

lock combination: 010

(B) (5 Points) Using the specification and clues from the partially completed diagrams above fill in the information that is missing from the state transition diagram and its accompanying truth table. When done:

- each state in the transition diagram should be assigned a 2-bit state name $S_1S_0$ (note that in this design the state name is not derived from the combination that opens the lock),
- the arcs leaving each state should be mutually exclusive and collectively exhaustive,
- the value for U should be specified for each state, and
- the truth table should be completed.

(complete above transition diagram and table)
Problem 3. (6 Points) The Mysterious XYZ Machine

An unidentified government agency has a design for a combinational device depicted below:

![Diagram of the combinational device]

Although you don’t know the function of each of the component modules, they are each combinational and marked with their respective propagation delays. You have been hired to analyze and improve the performance of this device.

NOTE: Scratch copies of the above diagram at the end of the quiz.

(A) (1 Point) What are the throughput and latency for the unpipelined combinational device?

Latency: \( 180 \) ns; Throughput: \( \frac{1}{180} \) ns\(^{-1} \)

(B) (4 Points) Show how to pipeline the above circuit for maximum throughput, by marking locations in the diagram where registers are to be inserted. Use a minimum number of registers, but be sure to include one on the output. Assume that the registers have \( 0 \) t\(_{PD} \) and t\(_{SETUP} \).

\[ \text{Goal: } t_{AX} = 60 \]  
(mark register locations in diagram above)

(C) (1 Point) What are the latency and throughput of your pipelined circuit?

Latency: \( 180 \) ns; Throughput: \( \frac{1}{60} \) ns\(^{-1} \)
Problem 4. (13 points): Software Reverse Engineering

You are given the following listing of a C program and its translation to Beta assembly code (shown to the right):

```c
int f(int x, int y) {
    int a = (x+y) >> 1;
    if (a == 0) return y;
    else return ???;
}
```

(Recall that $a >> b$ means $a$ shifted $b$ bits to the right, preserving the sign of $a$.)

(A) (3 points) In the space below, fill in the binary value of the BR instruction stored at the location tagged ‘yy:’ in the above program.

```
<table>
<thead>
<tr>
<th>BEq</th>
<th>LP = R2E</th>
<th>R31</th>
</tr>
</thead>
<tbody>
<tr>
<td>0010</td>
<td>00100111</td>
<td>01100</td>
</tr>
</tbody>
</table>
```

(fill in missing 1s and 0s for instruction at yy:)

(B) (1 point) Suppose the MOVE instruction at the location tagged ‘zz:’ were eliminated from the above program. Would it continue to run correctly?

Still works fine? Circle one: YES ... NO

(C) (2 points) Give the missing expression designated by ‘???’ in the C program above.

$$a + f(y, x-a)$$

(Write missing C expression)
The procedure $f$ is called from an external procedure and its execution is interrupted during a recursive call to $f$, just prior to the execution of the instruction tagged "bye: ". The contents of a region of memory are shown to the left.

**NB: All addresses and data values are shown in hex.** The BP register contains $0x250$, and SP contains $0x258$, and R0 contains $0x5$.

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>204:</td>
<td>CC</td>
</tr>
<tr>
<td>208:</td>
<td>4</td>
</tr>
<tr>
<td>20C:</td>
<td>7</td>
</tr>
<tr>
<td>210:</td>
<td>6</td>
</tr>
<tr>
<td>214:</td>
<td>7</td>
</tr>
<tr>
<td>218:</td>
<td>E8</td>
</tr>
<tr>
<td>21C:</td>
<td>D4</td>
</tr>
<tr>
<td>220:</td>
<td>8</td>
</tr>
<tr>
<td>224:</td>
<td>2</td>
</tr>
<tr>
<td>228:</td>
<td>1</td>
</tr>
<tr>
<td>22C:</td>
<td>6</td>
</tr>
<tr>
<td>230:</td>
<td>54</td>
</tr>
<tr>
<td>234:</td>
<td>??</td>
</tr>
<tr>
<td>238:</td>
<td>1</td>
</tr>
<tr>
<td>23C:</td>
<td>6</td>
</tr>
<tr>
<td>240:</td>
<td>3</td>
</tr>
<tr>
<td>244:</td>
<td>1</td>
</tr>
<tr>
<td>248:</td>
<td>54</td>
</tr>
<tr>
<td>24C:</td>
<td>238</td>
</tr>
</tbody>
</table>

**BP→ 250:** 3
**R1→ 254:** 3
**SP→ 258:** -1

(D) (1 point) What are the arguments to the currently active call to $f$?

Most recent arguments (HEX): $x=0x_{1__}; y=0x_{3}$

(E) (1 point) What value is at stored at location 0x234, shown as ??? in the listing to the left?

Contents $0x234$ (HEX): $0x_{220}$

(F) (1 point) What are the arguments to the original call to $f$?

Original arguments (HEX): $x=0x_{7__}; y=0x_{6}$

(G) (1 point) What value is in the LP register?

Contents of LP (HEX): $0x_{54}$

(H) (1 point) What value was in R1 at the time of the original call?

Contents of R1 (HEX): $0x_{8}$

(I) (1 point) What is the hex address of the instruction tagged "yy:"?

Address of $yy$ (HEX): $0x_{50}$

(J) (1 point) What value will be returned in R0 as the value of the original call?

Value returned to original caller (HEX): $0x_{14_{10}}$

END OF QUIZ!