Please enter your name and Athena login name in the spaces above. Mark the section where you would like to get your graded quiz on Wednesday. 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. Beta Implementation (9 points)

(A) (4 Points) The benchmark program shown at the right will be run on Betas with different stuck-at faults. A “stuck-at” fault happens when a signal is shorted to VDD or GND and is permanently a logic 1 or logic 0. For each of the faulty Betas listed below, please give the final value found in the location labeled by A when the benchmark reaches the HALT instruction. Assume that all registers start with the value 0.

RA2SEL stuck-at logic "0"
ST uses R0 instead of R1. Rb = 0, k bits [hs:il] of literal field
Final value found in location A: 1

ASEL stuck-at logic "0"
no affect on these instructions
Final value found in location A: 3

BSEL stuck-at logic "1"
breaks ADD instead of R1, 2nd ground is 0x100000
Final value found in location A: 0x801

WDSEL[1:0] stuck-at logic "0"
always store PC+4 in dest. reg.
Final value found in location A: 0x C

0 = 0
0 LD(R31, A, R0) R1 ← 1
q SHLC(R0, 1, R1) R1 ← 2
0 ADD(R0, R1, R1) R1 ← 3
0 ST(R1, A, R31) A ← 3
A: LONG(1)
(B) (5 points) For this problem assume that each register has been initialized to the value 0x00000000 where "??" is the register number as a two-digit hex number. So R0 is initialized to 0x00000000, R1 to 0x00000100, ..., and R30 to 0x00001E00. R31 of course always reads as 0.

For the SUBC instruction below, please indicate the values that will be found in the unpipelined Beta datapath just before the end of the clock cycle in which the instruction is executed. If the value doesn't matter since it's not used during the execution of the instruction or can't be determined, write "--". Otherwise enter the value in decimal, hex (use a "0x" prefix), or binary (use a "0b" prefix).

\[ . = 0x468 \]

**SUBC(R1, 0xFFF, R2)**

- RA1 input to register file: 0x100
- RA2 input to register file: --
- RD1 output from register file (JT): 0x100
- RD2 output from register file (MWD): --
- A input to ALU: 0x100
- B input to ALU: -1 = 0xFFF
- Output from ALU (MA): 0x101
- Output from Data Memory (MRD): --
- WD input to Register File: 0x101
- Input to the PC register (output of RESET mux): 0x46C
Problem 2. Caches (9 points)

(A) (1 Point) Consider a memory system with two levels of cache between the CPU and main memory. The access times and hit ratios are as shown.

- CPU → L1 Cache → L2 Cache → Main mem
  - Access time: 2 cycles
  - Hit ratio: 95%
  - Access time: 4 cycles
  - Hit ratio: 99%
  - Access time: 100 cycles

What is the average memory access time, in cycles, as seen by the CPU? You may express your answer as a formula if you wish.

\[
\text{Average memory access time (cycles)}: \frac{2.25}{\text{cycles}}
\]

Assume the L1 cache above is organized as a direct-mapped cache with 4 lines and block size of 2. A snapshot of the cache contents is shown on the right. Assume that any unspecified bits are 0 and that all cache entries are valid. D[0] and D[1] are the two data words on each cache line, with D[0] occupying the lower address in main memory.

<table>
<thead>
<tr>
<th>Line #</th>
<th>Tag</th>
<th>D[0]</th>
<th>D[1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x42</td>
<td>0x739F0083</td>
<td>0x73FF0121</td>
</tr>
<tr>
<td>1</td>
<td>0x17</td>
<td>0x73FF012C</td>
<td>0x619F0044</td>
</tr>
<tr>
<td>2</td>
<td>0x00</td>
<td>0x73FF0293</td>
<td>0x515F003C</td>
</tr>
<tr>
<td>3</td>
<td>0x42</td>
<td>0x627F0060</td>
<td>0x73FF023C</td>
</tr>
</tbody>
</table>

(B) (3 Points) The cache will divide the 32-bit address supplied by the CPU into three fields: B bits of block offset (including byte offset bits), L bits of cache line index, and T bits of tag field. Based on the cache parameters given above, what are the appropriate values for B, L, and T?

- value for B: 3
- value for L: 2
- value for T: 7

(C) (1 Point) Cache line 3 contains the data value 0x73FF023C. What is the address of this data value in main memory?

Address of 0x73FF023C in main memory: 0x85C

6.004 Spring 2018 - 3 of 5 - Quiz #3
Consider the short benchmark program shown to the right running using the L1 cache described earlier. Recall that the L1 cache is organized as a 4-line direct-mapped cache with a block size of 2.

\[
\begin{align*}
\text{.} & = 0 \\
\text{CMOVE(1000, R1)} & \quad \text{line} 0 \\
\text{LOOP:} & \\
\text{LD(R31, 0x114, R0)} & \quad \text{line} 1 \\
\text{LD(R31, 0x118, R0)} & \quad \text{line} 2 \\
\text{SUBC(R1, 1, R1)} & \quad \text{line} 3 \\
\text{BNE(R1, 0, LOOP)} & \\
\text{HALT()} & \\
\end{align*}
\]

(D) (1 point) There are 4 instructions in the loop (LD, LD, SUBC, BNE). If all the instructions are resident in the cache, which cache lines will they occupy? The four cache lines are numbered 0 through 3.

**Cache lines occupied by loop instructions:** 0, 1, 2

(E) (1 point) The two LD instructions access locations 0x114 and 0x118 during each iteration of the loop. If each data word is resident in the cache, which cache line will it occupy?

**Cache line occupied by data in location 0x114:** 2

**Cache line occupied by data in location 0x118:** 3

(F) (2 points) Compute the steady-state hit ratio for one iteration of the loop, i.e., after the benchmark has run for a while, say, the 10th iteration of the loop. Note that each iteration of the loop requires 4 instruction fetches and 2 data accesses.

**Steady-state hit ratio for one loop iteration:** \( \frac{4}{6} = 66.7\% \)

Loop has:
4 instruction fetches
2 data loads

access to 0x114 ⇒ replaces line 2 (collision)
fetch of BNE ⇒ replaces line 3 ⇒ 2 misses/iteration
access to 0x118 ⇒ hit in line 3
Problem 3. Pipelined Beta (12 points)

HKN has collected a bunch of Beta systems to offer at their annual fund-raiser, a used Beta yard sale. The Betas they have to sell fall into three categories:

F a fully-working 5-stage pipelined Beta presented in lecture, with full bypass paths, branch annulment (replacing the instructions that follow JMPs and taken branches with NOP), and stalling.

M A “minimalist” broken 5-stage pipelined Beta - like that shown in lecture, but with NO bypass paths, branch annulment, stalling.

B A broken 5-stage pipelined Beta, with full bypass paths, branch annulment, and stalling but with a defect in the register file. The register file works fine, except that R5 always reads as 0.

In order to identify which category each Beta falls into, they have run a simple benchmark on each Beta, and observe the values left in R0 and R1 when the program halts. The Beta benchmark is shown to the right. Assume that the initial value of each register is zero.

Note that the benchmark uses the macro NOP (an ADDC instruction that has no side effects). The NOPs are there to ensure that the result of prior instructions have been stored in the register file by the time the HALT is reached.

Your job is to figure out the values that will be left in R0 and R1 when this benchmark code is run on (possibly defective) Betas from each of the above categories. Please enter those values in the table to the right.

// Simple benchmark:
ADDCA(R31, 100, R5)  R5: 100 100 100
ADDCA(R5, 27, R0)    R0: 127 27 127
BR(A)                - taken -
MULCA(R5, 3, R0)
A: SUBCA(R5, 10, R1) R1: 90 50 -10
NOP() NOP() NOP()    reads from register
HALT()               

<table>
<thead>
<tr>
<th>Version</th>
<th>Value in R0</th>
<th>Value in R1</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>127</td>
<td>90</td>
</tr>
<tr>
<td>M</td>
<td></td>
<td>90</td>
</tr>
<tr>
<td>B</td>
<td>127</td>
<td>-100</td>
</tr>
</tbody>
</table>

Hint: It will be very helpful (but not required!) to fill in the pipeline diagram below showing the operation of version F. There are additional pipeline diagrams at the end of the quiz.

<table>
<thead>
<tr>
<th>IF</th>
<th>ADDC</th>
<th>ADDC</th>
<th>BR</th>
<th>MULC</th>
<th>SUBC</th>
<th>NOP</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF</td>
<td>ADDC</td>
<td>ADDC</td>
<td>BR</td>
<td>NOP</td>
<td>SUBC</td>
<td>NOP</td>
</tr>
<tr>
<td>ALU</td>
<td>ADDC</td>
<td>ADDC</td>
<td>BR</td>
<td>NOP</td>
<td>SUBC</td>
<td>NOP</td>
</tr>
<tr>
<td>MEM</td>
<td>ADDC</td>
<td>ADDC</td>
<td>BR</td>
<td>NOP</td>
<td>SUBC</td>
<td>NOP</td>
</tr>
<tr>
<td>WB</td>
<td>ADDC</td>
<td>ADDC</td>
<td>BR</td>
<td>NOP</td>
<td>SUBC</td>
<td></td>
</tr>
</tbody>
</table>

END OF QUIZ 3!

6.004 Spring 2018 - 5 of 5 - Quiz #3