Computer Architecture, Section 379: Homework #3

Yousef Alaa Awad

October 27, 2025

1

Given: Modify the single-cycle datapath (page 3/3) by implementing the instruction 'SWI' (store-word-and-increment). Below is the syntax. The instruction below will store the value in $t0 to memory location at address ($s0+88). Then, it will increment $s0 by 4.

SWI $t0, 88($s0)

This is the encoding of the instruction above:

unique	s0	t0	88
opcode (6)	rs (5)	rt (5)	offset (16)

Draw the changes on the datapath diagram and provide the values of all the control signals.

2

Given: Modify the single-cycle datapath by implementing the conditional move instruction 'MOVNZ' (move-if-not-zero). Below is the syntax. The instruction below will move $t1 into $t0 if $t2!=0. Otherwise, $t0 will be set to "0".

MOVNZ $t0, $t1, $t2

This is the encoding of the instruction above:

unique	t1	t2	t0	0	0
opcode (6)	rs (5)	rt (5)	rd (5)	shamt (5)	funct (6)

A) Draw the changes on the datapath diagram and provide the values of all the control signals.

3

Given: Consider a memory system with the following paramaters:

- Translation Lookaside Buffer has 512 entries and is 2-way set associative.
- 64KByte L1 Data Cache has 128 byte lines and is also 2-way set associative.
- Virtual addresses are 64 bits and physical addresses are 32 bits.
- 4KB page size.

```
quil@snowflake:HW/hw1 ‹main*›$ gcc main.c
quil@snowflake:HW/hw1 ‹main*›$ ./a.out
Printing Bits for Integer 0x12131415...
Byte 1: 0x15
Byte 2: 0x14
Byte 3: 0x13
Byte 4: 0x12

Printing Bits for Float 34.73...
Byte 1: 0x85
Byte 2: 0xEB
Byte 3: 0x0A
Byte 4: 0x42
quil@snowflake:HW/hw1 ‹main*›$
```

Listing 1: My C Code

```c
#include "stdio.h"
#include "stdint.h"

int main()
{
  // Variables
  int32_t sampleInt = 0x12131415;
  float sampleFloat = 34.73;
  // Pointer for individual finding of 2 bytes
  uint8_t *bytePointer = (uint8_t *)&sampleInt; // typecasting for funny reasons

  // Printing out the thing we are wanting
  if (printf("Printing Bits for Integer 0x12131415...\n") != 40)
  {
    return 1;
  }
  // Individually printing out 2 bytes at a time for Integer
  for (int i = 0; i < 4; i++)
  {
    if (printf("Byte %d: 0x%.2X\n", i+1, bytePointer[i]) != 13)
    {
      return 1;
    }
  }

  // Printing out the thing we are wanting x2
  if (printf("\nPrinting Bits for Float 34.73...\n") != 34)
  {
    return 1;
  }
  // Moving pointer to the float
  bytePointer = (uint8_t *)&sampleFloat;
  // Individually printing out 2 bytes at a time for Floats
  for (int i = 0; i < 4; i++)
  {
    if (printf("Byte %d: 0x%.2X \n", i+1, bytePointer[i]) != 14)
    {
      return 1;
    }
  }
  // Return success code
  return 0;
}
```