Computer Architectures: Homework 1

# Question 1

|  |
| --- |
| data ← DMEM**[**data\_addr**];** -- register indirect addr.  parity ← 0**;** -- initialize variables  i ← word\_length**;**  mask ← 1**;**  **while** i ≠ 0 do -- for each bit in word  temp ← data and mask**;** -- check if it is one  parity ← temp xor parity**;** -- keep track of the parity  data ← data **>>** 1**;** -- go to next bit  i ← i – 1**;**  **end** **while;**  DMEM**[**data\_addr**+**2**]** **<=** parity**;** -- base plus offset addr. |
|  |

## Extra instructions

The current instruction set for architecture B is lacking the XOR and AND instructions. These are both logic instructions so they will be grouped with the other logic instructions. This means there opcodes will start with 01 (the same as the other logic instructions). Both of these instruction take the three operands Rt, Ra and Rb and so will use the same instruction coding as the ADD and SUB instructions (ADD and SUB also take Rt, Ra and Rb operands).

|  |  |  |  |
| --- | --- | --- | --- |
| Command | Operands | Opcode | Fields |
| and | Rt, Ra, Rb | 01 100 | xxx xxxx xBBB xxxx xAAA xxxx xTTT |
| xor | Rt, Ra, Rb | 01 110 | xxx xxxx xBBB xxxx xAAA xxxx xTTT |

## Conversions

### ASSEMBLY

This is the above register transfer level code converted to assembly

|  |
| --- |
| -- data\_addr is in R7  -- word\_length is in R1    move R2, R7 -- R2 is data  movi R3, 0x0000 -- R3 is parity  move R4, R1 -- R4 is i  movi R5, 0x0001 -- R4 is mask  loop\_start\_label:  br R4, 000,loop\_end\_label -- if i is zero exit the loop  and R6, R2, R4 -- bit mask data  xor R3, R6, R3 -- keep track of parity  shr R2, R2, 1 -- go to next bit  dec R4, R4  br R0, 000, loop\_start\_label -- go back to the beginning of the loop    loop\_end\_label:  storo R7, R3, 2 -- store output |
|  |

## Machine Code – Binary

This is the above assembly code hand assembled into binary. Using ‘x’ as “don’t cares”

|  |
| --- |
| 1000 1xxx xxxx xxxx xxxx x111 xxxx x010 -- move R2, R7  1000 0xxx 0000 0000 0000 0000 xxxx x011 -- movi R3, 0x0000  1000 1xxx xxxx xxxx xxxx x001 xxxx x100 -- move R4, R1  1000 0xxx 0000 0000 0000 0001 xxxx x011 -- movi R5, 0x0001  -- loop\_start\_label:  1100 0000 0000 0110 xxxx x100 xxxx xxxx -- brz R4, loop\_end\_label  0110 0xxx xxxx x100 xxxx x010 xxxx x110 -- and R6, R2, R4  0111 0xxx xxxx x011 xxxx x110 xxxx x011 -- xor R3, R6, R3  0100 1xxx xxxx 0001 xxxx x010 xxxx x010 -- shr R2, R2, 1  0001 1xxx xxxx xxxx xxxx x100 xxxx x100 -- dec R4, R4  1100 0111 1111 1011 xxxx x000 xxxx xxxx –- br R0, 000, loop\_start\_label  -- loop\_end\_label:  1011 1000 0000 0010 xxxx x011 xxxx x111 -- storo R7, R3, 2 |
|  |

With “don’t cares” converted to 0’s.

|  |
| --- |
| 1000 1000 0000 0000 0000 0111 0000 0010 -- move R2, R7  1000 0000 0000 0000 0000 0000 0000 0011 -- movi R3, 000000  1000 1000 0000 0000 0000 0001 0000 0100 -- move R4, R1  1000 0000 0000 0000 0000 0001 0000 0011 -- movi R5, 000001  -- loop\_start\_label:  1100 0000 0000 0110 0000 0100 0000 0000 -- brz R4, loop\_end\_label  0110 0000 0000 0100 0000 0010 0000 0110 -- and R6, R2, R4  0111 0000 0000 0011 0000 0110 0000 0011 -- 0or R3, R6, R3  0100 1000 0000 0001 0000 0010 0000 0010 -- shr R2, R2, 1  0001 1000 0000 0000 0000 0100 0000 0100 -- dec R4, R4  1100 0111 1111 1011 0000 0000 0000 0000 -- jmp loop\_start\_label  -- loop\_end\_label:  1011 1000 0000 0010 0000 0011 0000 0111 -- storo R7, R3, 2 |
|  |

### Machine Code – Hex

This is the above binary machine code converted to hexadecimal. All “don’t cares” have been converted to zero so that the values can be represented in hex.

|  |
| --- |
| 0x88000702 -- move R2, R7  0x80000003 -- movi R3, 000  0x88000104 -- move R4, R1  0x80000103 -- movi R5, 001  -- loop\_start\_label:  0xC0060400 -- brz R4, loop\_end\_label  0x60040206 -- and R6, R2, R4  0x70030603 -- 0or R3, R6, R3  0x48010202 -- shr R2, R2, 1  0x18000404 -- dec R4, R4  0xC7FB0000 -- jmp loop\_start\_label  -- loop\_end\_label:  0xB8020307 -- storo R7, R3, 2 |
|  |

## Control signals

# Question 2