Computer Organization and Architecture Laboratory (CS39001)

Multi-cycle CPU Design

AIM: To design a multi-cycle CPU with the following ISA specification:

Consider a 16 bit stack based CPU with 8 GPRs `r0 -- r7`. The top entry of a memory resident stack pointed to by a CPU resident stack pointer `sp` provides one operand while the other operand is provided by the GPR specified in the instruction. The stack pointer `sp` gets initialized to the highest memory address on POWER ON or manual reset. The memory is word addressable where the word length is 2 Bytes. The ISA specification is as follows.

```
push rm     : sp <= sp-1; M[sp] <= rm;
pop  rm     : rm <= M[sp]; sp <= sp + 1;
add  rm     : rm <= rm + M[sp]; sp <= sp + 1;
neg  rm     : rm <= -M[sp]; sp <= sp + 1
                --- two's complement provides the negated value;
or   rm     : rm <= rm | M[sp]; sp <= sp + 1; --- bit-wise or
not  rm     : rm <= bit-wise complement of M[sp]; sp <= sp + 1;
   Corresponding to add, neg, or, not instructions, four flags are decided
   namely, carry (c), zero (z), overflow (v) and sign (s) based on the
   result of the operation; thus, after add operation, s <- 1 if the alu
   output is less than 0.
b<cc> label : if conditions[cc] = 1 then PC <= PC + label, where
              cc specifies the desired condition code to choose from the set
              {unconditional, carry (c), not carry (nc), zero (z),
               not zero (nz), overflow (v), not overflow (nv), sign (s),
               not sign (ns)} as decided by the most recently executed  ALU
               instruction; it is assumed that immediately after fetching of
               an instruction, PC is incremented (by 1 -- since there is no
               provision of accessing bytes).
call  label : sp <= sp - 1; M[sp] <= PC; PC <= PC + label; again, it is
              assumed that PC is incremented by 1 immediately after fetching
              the instruction.
ret         : PC <= M[sp]; sp <= sp + 1;
```

Assignment Statement:

1. Write a program segment in the above assembly language to take three values `a,` `b` and `c` from the stack top , computes the expression `a + b - c` and puts its value on the stack top.

2. *Instruction Format Design* — Give an instruction format ensuring that every instruction is contained in one word and the `label` field (for `branch` and `call` instructions) is at least 12 bit long.
   Write down the assembly language program written for part 1 in m/c code (in hex).

3. *Data Path Design* Assume that none of the registers — the GPRs, SP, PC or any other special purpose register used — has any provision other than reading its content or writing into it; thus, any other operations like incrementation, decrementation, shifting, etc. can only be carried out through the ALU. Also, *no more than two internal data buses are allowed in the data path*,

   (a) Give a neat diagram for a multi-cycle data path of the CPU; ensure that details such as the exact ALU operation for an ALU instruction or the exact condition desired for a branch instruction are kept hidden from the controller.

    (b) Tabulate the sequence of concurrent RTL micro-operations for the fetch phase and the execution phases of all the instructions; write the corresponding sequence of control signals in the table.

4. *Controller Design*
    (a) Depict by means of a neatly drawn ASM chart the controller beahviour. Special credit will be given if the behaviour achieves sharing of code segments over the instructions.
    (b) Give a Verilog behavioural encoding of the controller with a suitable test bench providing for the memory module holding (sequences of) instructions.

5. *Data Path Encoding*
    (a) A top level Verilog structural encoding of the Data Path.
    (b) A Verilog structural encoding of the register bank.
    (c) Verilog behavioural encoding of the individual registers in the register bank, and other registers such as, SP, PC and any other registers used.
    (d) A Verilog behavioural encoding of the ALU.
    (e) A Verilog structural encoding of the status detection circuits, storing modules (i.e., the flag FFs) and a Verilog behavioural encoding of the status condition selection circuit.

6. *Testing of CPU*
    (a) Verilog structural encoding of the Data Path and controller interface.
    (b) Use the test bench devised in 4b to test the interface.

Submissibles with marks:
TAs may put the lab record marking in the lab record too (in addition to putting in the marking table).

Lab day 1 –

1. The sample program segment of part 1 – in the lab record. **[5]**
2. Instruction format design in the lab record. **[10]**
3. The m/c code (in hex) of the assembly program of part 1 – in the lab record. **[5]**
4. Data path diagram and the table as indicated in assignment statement part 3 – in the lab record. **[10 + 15 = 25]**

Lab record submissions will be marked by the TAs against
(i) 100% max. marks — up to 3 PM, 12.10..'17 (Thursday),
(ii) 75% max marks — up to 3 PM, 13.10.'17 (Friday)
(iii) 50% max marks —up to 5 PM, 13.10.'17 (Friday) (iv) 25% max marks – after Friday

Please turn over

Lab Day 2 –

Neatly drawn ASM chart for the controller behaviour – in the lab record. **[15]**

Note that special credit will be given if provision for sharing of code segments is incorporated in the behavioural flow. So in the lab record you must explain how the submodules are identified and the ASM chart must be presented in the form of these submodules.

Special credit for code sharing: **[10]**

Lab record submissions will be marked by the TAs against

(i) 100% max. marks — up to 3 PM, 13.10..'17 (Friday),
(ii) 80% marks — up to 5 PM, 13,10.'17 (Friday)
(iii) 60% marks — for later submissions

Lab Days 3 & 4 –

Controller behavioural encoding in Verilog with test bench – to be uploaded on the Moodle site. **[25]**

submissions will be marked by the TAs against

(i) 100% max. marks — up to 11.55 PM, 20.10..'17 (Friday)
(ii) 90% max marks — up to 11.55 PM, 22.10.'17 (Sunday)
(iii) 75% max marks — up to 3 PM, 26.10.'17 (Thursday)
(iv) 60% max. marks — later submissions

Lab Days 5 & 6 [26.10.'17 & 27.10.'17]

Data Path Encoding with the test bench

submissions will be marked by the TAs against

1. A top level Verilog structural encoding of the Data Path with test bench. **[10]**
   100% max marks – 26.10.'17 – 11.55 PM
   80% max marks – 27.10.'17 – 11.55 PM
   60% max marks – later submissions

2. A Verilog structural encoding of the register bank. **[8]**
   100% max marks – 26.10.'17 – 11.55 PM
   80% max marks – 27.10.'17 – 11.55 PM
   60% max marks – later submissions

3. Verilog behavioural encoding of the individual registers in the register bank, and other registers such as, SP, PC and any other registers used. **[5]**
   100% max marks – 26.10.'17 – 11.55 PM
   80% max marks – 27.10.'17 – 11.55 PM
   60% max marks – later submissions

4. A Verilog behavioural encoding of the ALU. **[7]**
   100% max marks – 28.10.'17 – 5.00 PM
   80% max marks – 29.10.'17 – 11.55 PM
   60% max marks – later submissions

5. A Verilog structural encoding of the status detection circuits, storing modules (i.e., the flag FFs) **[10]**
   100% max marks – 28.10.'17 – 5.00 PM
   80% max marks – 29.10.'17 – 11.55 PM
   60% max marks – later submissions

   and

6. a Verilog behavioural encoding of the status condition selection circuit. **[5]**
   100% max marks – 28.10.'17 – 5.00 PM

80% max marks – 29.10.'17 – 11.55 PM
60% max marks – later submissions

7. Complete encoding of the CPU with test bench to test it against a suitably chosen small program. **[10]**

   100% max marks – 28.10.'17 – 5.00 PM
   90% max marks – 29.10.'17 – 11.55 PM
   70% max marks – 2.11.'17 – 5.00 PM
   60% for later submissions.