CPEG 222 Spring 2014 HW1 Solution

Bo Lu

March 18, 2014

Solution of Question 1:
The number of cycles $= 3$ sec $\times 2 \times 10^9$ cycles/sec $= 6 \times 10^9$ cycles

Solution of Question 2:
There are a number of reasons for the move towards RISC machines away from CISC. Some of them are:
1) Since early computers had limited memory capacities and were expensive, having a CISC instruction set enabled performing complex operations with very few instructions (encoded within a smaller memory size compared to a corresponding RISC program). Since then memories have got cheaper and there has been a lot of advances in the design of cache hierarchies that permit RISC machines work−around longer instruction sequences.
2) Writing a compiler to generate efficient code is easier for a RISC architecture than for a CISC architecture as the compiler can take advantage of a lot of registers provided by the RISC architecture than a CISC.
3) RISC instructions are more regular. Fixed instruction length makes it easier for decoding and pipelinining.

Solution of Question 3:
The reason of adding "arithmetic-shift-right" is to keep the sign of the number after shifting because the left most bit indicates the sign of the number. When shifting left there is no such requirement, therefore MIPS does not offer an "arithmetic-shift-left" opcode.

Solution of Question 4:
Part a:

```
int cFunction(int source, int i, int j){
  int target;
  printf("source is 0x%x and %lu bytes long\n",source,sizeof(source));
  source = source << (32−(i+j));
  target = ((unsigned int ) source) >> (32−j);
  printf("target is 0x%x and %lu bytes long\n",target,sizeof(target));
  return target;
}
```

Part b: $(i = 13, j = 7)$

```
sll $s1, $s1, 12  # source << (32−(i+j))
srl $s2, $s1, 25  # target = source >> (32−j)
```

Solution of Question 5:

```
sub $t0, $s2, $s3  # y - z
sub $t0, $t0, $s4  # y - z - q
add $s1, $t0, $s1  # x = x + y - z - q
```

Solution of Question 6:

```
slt $t0, $s1, $s2  # t0 = 1 if $s1 < $s2
beq $t0, $zero, L1  # if $s1 >= $s2 goto L1
add $s0, $s2, $zero  # $s0 = $s2
j L2
L1: add $s0, $s1, $zero  # $s0 = $s1
L2: ...
```

Solution of Question 7:

```
Loop: sll $t0, $s1, 2  # i *= 4
add $t0, $t0, $s0  # align it A[i]
lw $t1, 0($t0)  # load word t1 = A[i]
slt t2, s3, t1  # t2 = 1 if A[i] > b
beq t2, zero, L2  # if A[i] <= b, exit the loop
add s1, s1, s2  # i=i+j
j Loop
L2:...
```