Coursework for Tutorial 2

Computer Architecture II:

November 1, 2023

1 Assignment

Task #1 Consider the C code shown in Figure 1. The function `min5` computes the minimum between 5 input integers by relying on a function that computes the minimum between 3 input integers (`min3`). Translate the code (following its implementation logic) in RISC-1, using the calling convention described in the Tutorial 2 slides. First produce an unoptimised version of the codes with NOPs. Then see whether it can be optimised and/or give justification for where it can't. For the printing of characters to output, you can assume the existence of an external function `printChar` that takes as input an ASCII coded characters and prints it to standard output.

```c
#include <stdio.h>
int min3(int a, int b, int c){
    int m = a;
    if(b<m)
        m = b;
    if(c<m)
        m = c;
    return m;
}
int min5(int a, int b, int c, int d, int e){
    return(min3(min3(a,b,c),d,e));
}
int main()
{
    int a = 100;
    int b = 89;
    int c = 65;
    int d = 12;
    int e = 32;
    int out;
    out = min5(a,b,c,d,e);
    if(out==a) printf("1");
    if(out==b) printf("2");
    if(out==c) printf("3");
    if(out==d) printf("4");
    if(out==e) printf("5");
    return 0;
}
```

Figure 1: C implementation of max4 function.

Task #2 Consider the 3-way ackermann function and its C implementation shown in Figure 2. Assume that the code is being compiled for a RISC-1 implementation with 8 register banks. Engineer

the C code so that it will compute and print to standard output:

1. the number of function calls to `ack3way`
2. the number of register overflow occurrences.
3. the number of register underflow occurrences.

Describe, also by using drawings of the register banks and the various pointers involved, how your code works.

Notice that you are not asked to translate the function in RISC-1. And it is humanly unfeasible to do the computation by hands for the m,n and p parameters used in the main call in Figure 2. You will have to run the programme on the following three input sequences:

1. $(m = 2, n = 3, p = 3)$
2. $(m = 1, n = 8, p = 8)$
3. $(m = 10, n = 5, p = 2)$.

Remember that at least two register banks need to be allocated at any one time.

```
int ack3way(int m, int n, int p)
{
    if(p==0)
        return m+n;
    if(n==0 && p==1)
        return 0;
    if(n==0 && p==2)
        return 1;
    if(n==0)
        return m;
    else
        return ack3way(m,ack3way(m,n-1,p),p-1);

}
int main()
{
    printf("Result: %d\n", ack3way(2,3,3));
    return 0;
}
```

Figure 2: C implementation of the function for Task #2.

2 Instructions

- For the coding tasks, submit the source code files. For Task2 also submit screenshot proof of the results you obtain on the three sequences of input provided. The register bank explanation can be either hand-drawn or done electronically, but must be readable! Submit everything in a single zip file.
- Due date: November 12 @ 12.59pm.
- This tutorial will count toward 10% of your final grade. Each task will count toward 50% of the overall tutorial score. Tasks 1 and 2 will be evaluated according to the rubric provided.