COMPUTER SCIENCE TRIPOS Part II – 2024 – Paper 8

1 Advanced Computer Architecture (swm11)

(*a*) For an out-of-order superscalar processor, what are false dependencies on register names and what hardware technique is often used to remove them? [4 marks]

(*b*) Why do out-of-order superscalar processors use a store queue (sometimes called a store buffer) whereas simple scalar processors do not? [4 marks]

(*c*) For an out-of-order superscalar processor, what are the trade-offs between using a reorder buffer and a unified register file to hold computed register values? [4 marks]

(*d*) Consider the following C-code implementation of bubble sort:

```
void bubbleSort(int array[], int size) {
  for (int step = 0; step < size - 1; ++step) {
    for (int i = 0; i < size - step - 1; ++i) {
      if (array[i] > array[i + 1]) {
        int temp = array[i];
        array[i] = array[i + 1];
        array[i + 1] = temp;
      }
    }
  }
}
```

(*i*) The 32-bit ARM ISA allows almost all instructions to be conditional (or *predicated*) whereas the newer 64-bit ARM ISA does not. Using the above code as an example, how could predicated execution be used to avoid data-dependent branch misprediction using if-conversion? [4 marks]

(*ii*) Why do modern out-of-order superscalar processors avoid predicated execution? Why might the above code result in a lot of branch mispredictions? [4 marks]