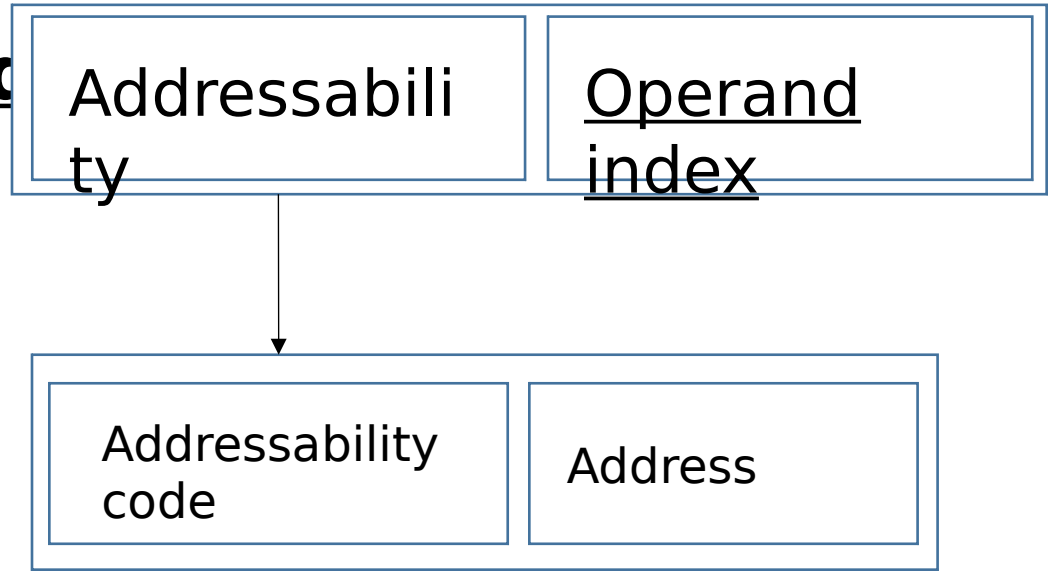
Code generation using multiple registers

To use multiple registers we need to modify our operand descriptor format like the following:

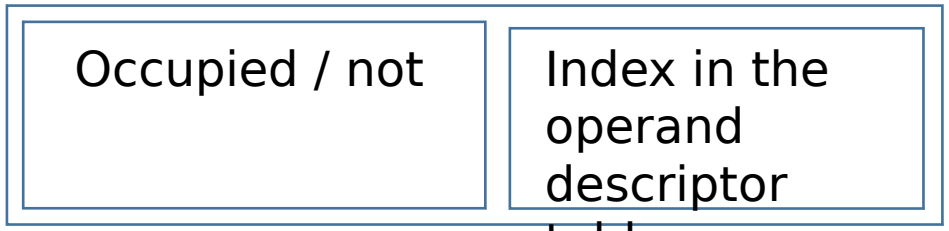
The new field **‘operand ind**

stores its own index in the

Operand descriptor table.



REGISTER DESCRIPTOR



A register descriptor is created when a new register is required to store the value of the intermediate result.

The first field indicates whether the register is free or not and the second field contains the index in the operand descriptor table. This is the index which contains the operand descriptor which represents the operand whose value is stored in this register.

A register descriptor table is also created which contains register descriptors of the above mentioned form.

The modified functions (to utilize multiple registers)

```
build_descriptor (opd1);
{
    i= i + 1;
    operand_descr[i]=(type, addressibility, address) of opd1;
    if (addressability code='R')
    {
        reg_top++;
        reg_descriptor[reg_top].occupied='YES';
        reg_descriptor[reg_top].operand_index=i;
     }
    return i;
  }
// i and reg_top are 0 initially.
```

The modified functions (Cont.)

```
codegen (operator, opd1, opd2)
{
  if  opd1.addressibilty =‘R’
  {
          if operator = ‘*’ generate(‘MULT opd1.address, opd2’);
          else ……..
          return opd1.operand_index;
  }
    else  if    opd2.addressibilty =‘R’
  {
          if operator = ‘*’ generate(‘MULT opnd2.address, opd1’);
          else ……
         return opd2.operand_index;
  }
     else  {
            temp=build_descriptor(new reg operand);        // Creating a new register operand
            generate (‘LOAD temp,opd1’);
            if operator = ‘*’ generate(‘MULT operand_des[temp].address, opd2’);
            else……
            return temp;
        }
}
```