```
#ifndef MEMORY_H
#define MEMORY_H 1

#include <iostream.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>

class Memory {
private:
    int mem_array[256];
public:
    // default constructor for Machine objects
    Memory();
    ~Memory();
    void Load_Memory(int, int);
    void Dump_Mem(fstream&);
    int Get_Memory(int);
    int Get_OP(int);
    int Get_U2(int);
    int Get_R(int);
    int Get_X(int);
    int Get_U1(int);
    int Get_S(int);
};

Memory::Memory()
{
    //Default Consructor definition, set all memory addresses w/ a
    //halt & dump instruction.
    int n = 0;
    while(n < 256)
    {
    mem_array[n] = 0xC0C0;
    n++;
    }
}
Memory::~Memory()
{
    // Default Destructor definition
}

//Public Operations

void Memory::Load_Memory(int mem_location, int mem_value)
{
    mem_array[mem_location] = mem_value;
}

void Memory::Dump_Mem(fstream& outs)
{
    outs << "\nMemory Values:\n";
    int n = 0;
    while ((n+7) < 256)
    {
    outs << dec  << n << ':' << hex << mem_array[n] << ' '
         << dec  << (n+1) << ':' << hex << mem_array[(n+1)] << ' '
         << dec  << (n+2) << ':' << hex << mem_array[(n+2)] << ' '
         << dec  << (n+3) << ':' << hex << mem_array[(n+3)] << ' '
         << dec  << (n+4) << ':' << hex << mem_array[(n+4)] << ' '
         << dec  << (n+5) << ':' << hex << mem_array[(n+5)] << ' '
         << dec  << (n+6) << ':' << hex << mem_array[(n+6)] << ' '
         << dec  << (n+7) << ':' << hex << mem_array[(n+7)] << '\n';
    //printf("%d:%5X ", (start + n), mem_array[(start + n)]);
    n += 8;
    }
    outs << '\n';
}
```

```
int Memory::Get_Memory(int mem_location)
{
    return mem_array[mem_location];
}

int Memory::Get_OP(int mem_location)
{
    int OP = Get_Memory(mem_location), op_shifter = (int)pow(2,16);
    OP /= op_shifter;
    return OP;
}

int Memory::Get_U2(int mem_location)
{
    int U2 = Get_Memory(mem_location), temp = U2, junk_shifter = (int)pow(2,16),
    u2_shifter = (int)pow(2,14);
    temp /= junk_shifter;
    temp *= junk_shifter;
    U2 -= temp;
    U2 /= u2_shifter;
    return U2;
}

int Memory::Get_R(int mem_location)
{
    int R = Get_Memory(mem_location), temp = R, junk_shifter = (int)pow(2,14), r_shifter =
     (int)pow(2,12);
    temp /= junk_shifter;
    temp *= junk_shifter;
    R -= temp;
    R /= r_shifter;
    return R;
}

int Memory::Get_X(int mem_location)
{
    int X = Get_Memory(mem_location), temp = X, junk_shifter = (int)pow(2,12), x_shifter =
     (int)pow(2,10);
    temp /= junk_shifter;
    temp *= junk_shifter;
    X -= temp;
    X /= x_shifter;
    return X;
}

int Memory::Get_U1(int mem_location)
{
    int U1 = Get_Memory(mem_location), temp = U1, junk_shifter = (int)pow(2,10),
    u1_shifter = (int)pow(2,8);
    temp /= junk_shifter;
    temp *= junk_shifter;
    U1 -= temp;
    U1 /= u1_shifter;
    return U1;
}

int Memory::Get_S(int mem_location)
{
    int S = Get_Memory(mem_location), temp = S, junk_shifter = (int)pow(2,8);
    temp /= junk_shifter;
    temp *= junk_shifter;
    S -= temp;
    return S;
}

#endif
```