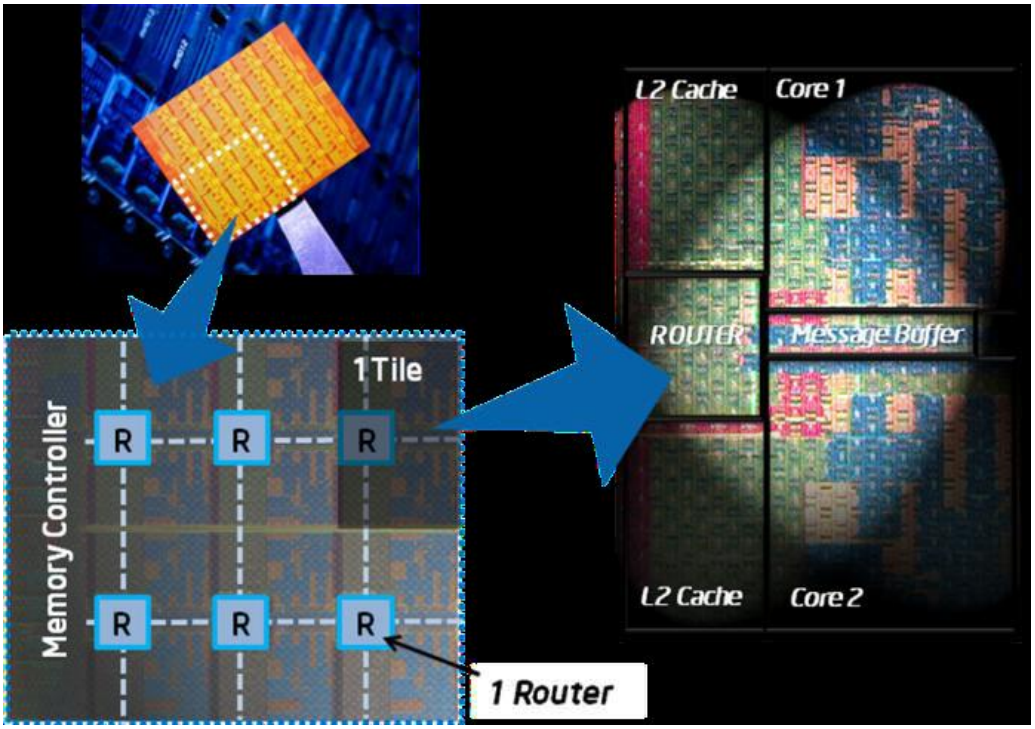
Hacettepe Üniversitesi
Bilgisayar Mühendisliği Bölümü
BİL431 Koşut İşlem
2. Ara sınav 18-12-2009

Soru 1. "Barrier" işlevini, iki boyutlu "Torus" topolojisi için gerçekleştiren algoritmayı tasarlayınız.

Soru 2. "MPI_Barrier" işlevini, iki boyutlu "Torus" topolojisi için, diğer MPI işlevlerini kullanarak C programlama dili ile kodlayınız.

Soru 3. "Barrier" algoritmanızı, Intel tarafından bu ay tanıtımı yapılan "Single-chip Cloud Computer," (SCC) araştırma mikroişleyicisinde nasıl çalıştırabileceğinizi tartışınız.



Anatomy of the Single-chip Cloud Computer

Intel Labs has created an experimental "Single-chip Cloud Computer," (SCC) a research microprocessor containing the most Intel Architecture cores ever integrated on a silicon CPU chip – 48 cores. It incorporates technologies intended to scale multi-core processors to 100 cores and beyond, such as an on-chip network, advanced power management technologies and support for "message-passing."

...

The name "Single-chip Cloud Computer" reflects the fact that the architecture resembles a scalable cluster of computers such as you would find in a cloud, integrated into silicon.

The research chip features:

- *24 "tiles" with two IA cores per tile*
- *A 24-router mesh network with 256 GB/s bisection bandwidth*
- *4 integrated DDR3 memory controllers*
- *Hardware support for message-passing*

Kullanabileceğiniz MPI işlevleri:

```
int MPI_Init(int *argc, char ***argv)
int MPI_Finalize()
int MPI_Comm_size(MPI_Comm comm, int *size)
int MPI_Comm_rank(MPI_Comm comm, int *rank)


int MPI_Send(void *buf, int count, MPI_Datatype datatype,int dest,
                  int tag, MPI_Comm comm)
int MPI_Recv(void *buf, int count, MPI_Datatype datatype,int source,
                  int tag, MPI_Comm comm, MPI_Status *status)
int MPI_Sendrecv(void *sendbuf, int sendcount,MPI_Datatype
                  senddatatype, int dest, int sendtag,void *recvbuf,
                  int recvcount, MPI_Datatype recvdatatype,int
                  source, int recvtag, MPI_Comm comm,MPI_Status
                  *status)

int MPI_Get_count(MPI_Status *status, MPI_Datatype datatype,int
                  *count)

int MPI_Cart_create(MPI_Comm comm_old, int ndims, int *dims, int
                  *periods, int reorder, MPI_Comm *comm_cart)
int MPI_Cart_rank(MPI_Comm comm_cart, int *coords, int *rank)
int MPI_Cart_coord(MPI_Comm comm_cart, int rank, int maxdims,  int
                  *coords)
```

`int MPI_Barrier(MPI_Comm comm)`
The only argument of MPI_Barrier is the communicator that defines the group of processes that are synchronized. The call to MPI_Barrier returns only after all the processes in the group have called this function.

Kullanabileceğiniz değişmezler: **`MPI_COMM_WORLD,MPI_ANY_SOURCE,MPI_ANY_TAG`**