

1 Hardware

Presently there are four CPUs in the Cray each of which has a peak performance of 133 MFlops. The Cray is 64-bit real memory machine. There are 128 MWords (1 GigaByte) of real memory available. Currently there are 15 GBytes of disk space, 2.5 GBytes of which are reserved for temporary file storage. There are no other printing or storage facilities available to user.

SYSTEM SPECIFICATIONS

CPU

CMOS Technology

30ns clock period

64 bit architecture

Maximum of 4 single processors or 8 dual processors

133 Mflops peak per CPU

Shared registers for inter-processor communication and synchronization

MEMORY

Shared central memory

4 ports per CPU

70 ns CMOS DRAM

256 - 1024 Mbytes memory (32 - 128 Mwords)

1.05 Gb/sec memory bandwidth per CPU, approximately 130 Mwords/sec

4.2 Gb/sec total bandwidth

64 memory banks (regardless of memory size)

I/O

VME based - 40 Mb/sec per VME

1 - 4 per CPU

1 - 4 in-cabinet (40 Gb maximum disk capacity)

Up to 3 expansion cabinets with a maximum of 16 VME sub-systems (200 Gb maximum disk capacity)

1.05 Gb/sec total system I/O

640 Mb/sec total VME bandwidth

100 Mb/sec Hippi

264 Mb/sec per CPU

PHYSICAL CHARACTERISTICS

635 Kg

1 square metre footprint

127 cm x 381 cm

6 kw power consumption

Air cooled

Operating temperature 10xb0 - 30xb0 C - any air-conditioned office is suitable - ie. an expensive controlled environment is not necessary

CPU SCHEMATIC

Registers

Primary registers accessible directly by the functional units - address (a), scalar (S) and vector (V).

V - 8 registers each containing 64 elements of 64 bits.

S - 8 registers (64 bits each).

A - 8 registers.

Intermediate registers B and T support (caching) for primary registers.

VM - 64 bit vector mask. Each bit refers to an element in a vector register. The mask uses merge and test instructions to select elements from a vector.

VL vector length values in the range 1 to 64 specifying the length of the vector to be processed by the vector instruction.

RTC - run time clock

1.2 Software

The operating system on the Cray is UNICOS (Cray version of Unix). All of the normal Unix commands are available. The available software is:

- * Introduction to Unicos for Application Programmers (iuap) - a computer based training package for new users of the Cray (see section 1.7.3);
 - * Compilers for Fortran 77, C and Pascal;
 - * Cray Tools - a suite of tools for measuring and improving the performance of programs run
- Application packages - PVM/Hence, Nag Libraries, Unichem, TurboKiva, MPGS.

1.3 Documentation

A full set of the Cray user documentation is available to users in Room 1028 of the David Bates building. Due to the verbose nature of Cray manuals, however, users are encouraged to make full use of this document and the on-line information facilities as described later on.

1.4 Logging on

The system name of the Cray is sn5227 (143.117.14.16). From any machine connected to the campus network the following command should allow access to the Cray:

```
telnet sn5227
```

For those not connected to the campus network the Cray can be accessed from the Vax using the same command.

1.5 X-windows

Many of the software utilities available on the Cray have optional X-windows interfaces. On a terminal running X-windows, OpenWindows or Motif these can be used to enhance the efficiency of the Unicos environment. X-windows is enabled by typing:

```
xhost +sn5227 (on your workstation)
```

setenv DISPLAY workstation.internet.number:0.0 (on the Cray)

To test that the X-windows connection has been set up correctly try starting up the X-windowed interface to the manual pages. Type

```
xman &
```

(An ampersand puts the task into the background leaving your login shell free).

This tool will give information on other X utilities. Users are encouraged to use the X-windows interfaces to the code performance and analysis tools.

1.8 On-line information

Three powerful utilities are provided on-line for users to learn about various aspects of Unicos and the Cray. User are encouraged to try these utilities before resorting to using the manuals.

1.9 Manual pages

For each Unicos command, a description, including all possible options, can be displayed on screen by typing:

```
man command_name
```

Also the '-i' and '-k' options of the 'man' command allow keyword searches.

2.0 Document viewer

Some Unicos manuals are held on-line in electronic form and can be accessed using a viewer called docview. Simply type:

```
docview
```

This brings up an initial command menu which is self-explanatory. Selecting the 'a' option gives a complete list of the manuals available. In particular the 'find' option is useful when searching for material.

2.1 Computer based training

There is a computer based training package available on the Cray called "Introduction to Unicos for Application Programmers". This is a excellent utility which contains material on all the important topics for new users of the Cray. Users are urged to study and make use of this facility. It is invoked by typing:

```
iuap
```

Again navigating around the package is self explanatory. The following topics are covered by the package:

The Unicos Environment

The Fortran Environment

The Cray Standard C Environment

Software Support Tools

Networking

Batch Processing

Each topic contains a number of lessons which cover the basic skills required to make use of the Cray.

2.2 explain & whatis

The commands `explain` and `whatis` can be used to obtain more information in certain situations:

If an operation terminates and gives an error code, eg fmp-59 which is an error from the Fortran multitasking pre-processor then the `explain` command will display text specific to that error when the user types the following:

```
explain fmp-59
```

The `whatis` command displays a summary of a command eg typing:

```
whatis ls
```

produces the following

```
ls(1) - Lists contents of directory
```

2.3 Running programs

This section contains general advice and guidance on compiling and running codes on the Cray.

2.4 Scratch disk - temporary file space

The Cray provides approximately 2.5 GigaBytes of temporary high speed access file space. This space can be accessed at a sustained rate of 6 MegaBytes per second (as compared to 2 MegaBytes per second for ordinary file space) and so should be used by programs which perform a large amount of file I/O.

Files can be written to the scratch disk by using the `assign` statement to assign a file in the `/tmp` directory to the relevant unit number eg

```
assign -a /tmp/gh09.dat u:05
```

The `/tmp` directory is "cleaned" on Friday evenings. At that time files which are older than seven days will be deleted. Users are however requested to remove files from `/tmp` as soon as possible after creation so that other users are able to generate large files.

2.5 The batch environment

The Cray at Queen's is primarily intended for use as a batch environment. Users should therefore, in general, submit all compute intensive operations to the batch queues. This includes compilation of large programs. Full instruction on how to use the batch system is given in the `iuap` package. The following is a brief summary.

To submit a job to a queue a script file must be written. The following is a very simple example of a script file which puts an executable called `spcc` in a subdirectory called `wamm` on to the queue. The `ja` lines cause some basic job accounting information to be provided in an output file after the execution of the program

```
#!/bin/csh
```

```
# QSUB-s /bin/csh
```

```
cd /home_a/ccg0521/wamm
```

```
ja
```

```
./spcc
```

```
ja -clst
```

This script file is then submitted to the queue system using the `qsub` command - which validates all

Network Queuing System request options. For example if the script file shown above is called 'runspcc' then type:

```
qsub runspcc
```

Using the '-q' option of the 'qsub' commands allow users to submit the job to a particular pipe queue on the Cray. There are four pipe queues each of which has three batch queues.

Users submit jobs to pipe queues only and the system automatically decides which batch queue is most suitable. The pipe queues, batch queues and their associated limits are shown in the following table:-

Queue Specifications of the QUB CRAY Y-MP EL

Pipe Queues	Batch Queues	CPU Limits (Secs)	Memory Limits (MMts)
small	smalls	36000	2
	mediums	36000	5
	larges	36000	10
medium	smallm	36000	12
	mediumm	36000	14
	largem	36000	16
large	smalll	36000	20
	mediuml	36000	40
	largel	36000	80
verylarge	verylarges	40000	90
	verylargel	40000	100

(Note: the limits on the queues may be varied slightly).

Users should note that the smaller queues run at higher priority. Jobs which are submitted to queues with much higher memory allocations than required by the job will therefore take longer to execute that necessary.

The 'qstat' command shows, for the batch queues, how many jobs are currently in each queue. It also shows, for each queue, the limit on the number of jobs from that queue allowed to run concurrently. For some of the larger queues this is set to 1 so that excessive swapping of jobs does not occur.

2.6 Using the NAG libraries

Versions of the NAG routines which have been optimized for the Cray are available. They can be linked into programs as follows:

```
cf77 -Wl"-l /usr/local/lib/libnag.a" prog.f
```

For information on specific routines, type 'naghel'. The Cray is 64-bit machine and so single precision on the Cray is equivalent to double precision on most other machines. The version of NAG libraries provided on the Cray is therefore single precision and this is denoted by the last letter of the routines being 'E'. If a routine name is listed in 'naghel' as ending in 'F' this should be replaced by 'E'.

2.7 Reducing compilation time - makefiles

Many user written codes running on the Cray consist of large source code files, and consequently during the development cycle, when small changes are being made to the code and the code recompiled, compilation takes a great deal of time. Compilation times can be vastly reduced by splitting up the source code and only recompiling the routines which have been altered. Unicos provides a fully automatic utility for performing the splitting of programs called 'fngen'.

For example to split a program contained in a file called 'modulate.f' the following should be typed:

fmgen modulate.f

It is advisable to have the program in a directory ie so there is only one file called makefile as this creates a separate source file for every procedure in the program and a `makefile' for automatic compilation of the program. When changes are made to any of the source code simply typing `make' in that directory brings the executable up to date by only compiling those pieces of source code which have been altered.

Example

File prog is generated by compiling and loading the main program in main.f and subroutines in sub1.f and sub2.f. To use the make utility to maintain this module first create a file named *makefile* containing the following:

```
prog: main.o sub1.o sub2.o
```

```
<tab> segldr -o prog main.o sub1.o sub2.o
```

The first line describes the dependencies ie prog depends on main.o - note that prog depends on the object files not the source files. The second line gives the rules for making a module from the object files.

When *make* is invoked to generate prog it firstly checks whether the files make depends on are up-to-date by searching the current directory for a source file corresponding to an object file and examing the modification dates etc. If the object file is older than the source it recompiles otherwise it assumes that recompilation is not necessary. *Make* then uses the rule in the second line to produce a new version of prog. Typing *make* when none of the .o files exist results in the files being automatically compiled and invokes segldr to load them.

2.8 File compression utility

A utility for compressing and uncompressing files is available on the Cray. This can be useful for reducing long term storage needs or for cutting down file transfer times between Cray systems. For example typing:

```
pack bigdatafile.dat
```

will result in a file called `bigdatafile.dat.z' being created in place of `bigdatafile.dat' the new file being up to 40% smaller than the original. Use `unpack' to recreate the original file.

Note: no packing occurs if the file name has more than 12 characters.