

# Intel IPP

## Intel Integrated Performance Primitives

Intel Integrated Performance Primitives, version 7.1.1, compiled for AVX vector instructions is available, via module ipp. The IPP is a very rich library of highly optimized algorithmic building blocks for media and data applications. This includes signal, image and frame processing algorithms, such as FFT, FIR, Convolution, Optical Flow, Hough transform, Sum, MinMax, as well as cryptographic functions, linear algebra functions and many more.

Check out IPP before implementing own math functions for data processing, it is likely already there.

```
$ module load ipp
```

The module sets up environment variables, required for linking and running ipp enabled applications.

## IPP example

```
#include "ipp.h"
#include <stdio.h>
int main(int argc, char* argv[])
{
    const IppLibraryVersion *lib;
    Ipp64u fm;
    IppStatus status;

    status= ippInit();          //IPP initialization with the best optimization layer
    if( status != ippStsNoErr ) {
        printf("IppInit() Error:n");
        printf("%sn", ippGetStatusString(status) );
        return -1;
    }

    //Get version info
    lib = ippiGetLibVersion();
    printf("%s %sn", lib->Name, lib->Version);

    //Get CPU features enabled with selected library level
    fm=ippGetEnabledCpuFeatures();
    printf("SSE      :%cn", (fm>1)&1?'Y':'N');
    printf("SSE2     :%cn", (fm>2)&1?'Y':'N');
    printf("SSE3      :%cn", (fm>3)&1?'Y':'N');
    printf("SSSE3     :%cn", (fm>4)&1?'Y':'N');
```

```

printf("SSE41   :%cn", (fm>6)&1?'Y':'N');
printf("SSE42   :%cn", (fm>7)&1?'Y':'N');
printf("AVX     :%cn", (fm>8)&1?'Y':'N');
printf("AVX2    :%cn", (fm>15)&1?'Y':'N' );
printf("-----n");
printf("OS Enabled AVX :%cn", (fm>9)&1?'Y':'N');
printf("AES          :%cn", (fm>10)&1?'Y':'N');
printf("CLMUL        :%cn", (fm>11)&1?'Y':'N');
printf("RDRAND       :%cn", (fm>13)&1?'Y':'N');
printf("F16C         :%cn", (fm>14)&1?'Y':'N');

return 0;
}

```

Compile above example, using any compiler and the ipp module.

```

$ module load intel
$ module load ipp

```

```

$ icc testipp.c -o testipp.x -lippi -lipps -lippcore

```

You will need the ipp module loaded to run the ipp enabled executable. This may be avoided, by compiling library search paths into the executable

```

$ module load intel
$ module load ipp

```

```

$ icc testipp.c -o testipp.x -Wl,-rpath=$LIBRARY_PATH -lippi -lipps -lippcore

```

## Code samples and documentation

Intel provides number of Code Samples for IPP, illustrating use of IPP.

Read full documentation on IPP on Intel website, in particular the IPP Reference manual.