72.20 Stack Manipulation Macros

dMARK

Declare a stack marker variable, `mark`, for the XSUB. See `MARK` and `dORIGMARK`.

```
                dMARK;
```

dORIGMARK

Saves the original stack mark for the XSUB. See `ORIGMARK`.

```
                dORIGMARK;
```

dSP

Declares a local copy of perl's stack pointer for the XSUB, available via the `SP` macro. See `SP`.

```
                dSP;
```

EXTEND

Used to extend the argument stack for an XSUB's return values. Once used, guarantees that there is room for at least `nitems` to be pushed onto the stack.

```
        void    EXTEND(SP, int nitems)
```

MARK

Stack marker variable for the XSUB. See `dMARK`.

mPUSHi

Push an integer onto the stack. The stack must have room for this element. Handles 'set' magic. Does not use `TARG`. See also `PUSHi`, `mXPUSHi` and `XPUSHi`.

```
        void    mPUSHi(IV iv)
```

mPUSHn

Push a double onto the stack. The stack must have room for this element. Handles 'set' magic. Does not use `TARG`. See also `PUSHn`, `mXPUSHn` and `XPUSHn`.

```
        void    mPUSHn(NV nv)
```

mPUSHp

Push a string onto the stack. The stack must have room for this element. The `len` indicates the length of the string. Handles 'set' magic. Does not use `TARG`. See also `PUSHp`, `mXPUSHp` and `XPUSHp`.

```
        void    mPUSHp(char* str, STRLEN len)
```

mPUSHu

Push an unsigned integer onto the stack. The stack must have room for this element. Handles 'set' magic. Does not use `TARG`. See also `PUSHu`, `mXPUSHu` and `XPUSHu`.

```
        void    mPUSHu(UV uv)
```