Macros in entry.S

macro MOD_INTEGER2BITMAP dst, src, tmp, mod

```
dst = 1 << sanitized(src) = 1 << (src && (mod - 1));
```

macro MOD_SHIFT2LEFT dst, tmp, mod

Move 1 bit left logically

```
dst = (dst >> (mod - 1)) | (dst << 1); // >> (mod - 1) means clean it
dst &= 1 << (mod -1)
```

macro MOD_SHIFT2RIGHT dst, tmp, mod

Move 1 bit right logically

```
dst = (dst << (mod - 1)) | (dst >> 1);
dst &= 1 << (mod - 1);
```

macro RESTORE_PSR r0

```
mv r0 to %psr
WR_delay
```

macro PSR_SET_ET_PIL r

```
%psr |= (PSR_ET_BIT|PSR_PIL_MASK);
```

macro PSR_UNSET_ET

```
%psr &= ~(PSR_ET_BIT)
```

macro PSR_UNSET_ET_PIL

```
%psr &= ~(PSR_ET_BIT|PSR_PIL_MASK)
```

macro GET_CPUID r

%asr17 is PCR;

bit 31 to 28 is Processor index

SRA Shift right arithmetic

```
#ifdef CONFIG_SMP
return %asr17 >> 28
#else
return 0
#endif
```

macro GET_CKTHREAD r, tmp

ckthread is stored in localSchedInfo[]

```
#ifdef CONFIG_SMP
tmp = GET_CPUID();
r = (localSched_t *)localSchedInfo() + tmp;
//r = localSchedInfo() + CpuId 6 * 4 = localSchedInfo() + CpuId * sizeof(localSched_t);
#else
r = localSchedInfo();
#endif

r = (char *)r + _CKTHREAD_OFFSET;
```

macro GET_CPUCTXT r, tmp

```
r = GET_CKTHREAD();
r = (char *)r + (_CTRL_OFFSET+_IRQCPUCTXT_OFFSET)
```

macro SET_CPUCTXT ctxt, tmp, tmp2

```
tmp = GET_CKTHREAD();
*((char *)tmp + (_CTRL_OFFSET+_IRQCPUCTXT_OFFSET) ) = ctxt
```

macro GET_CKTHREAD_STACK r t

```
r = GET_CKTHREAD();
t = CONFIG_KSTACK_SIZE - 8;
r = (char *)r + t;
```

macro GET_CKTHREAD_TBR r tmp

```
r = GET_CKTHREAD();
r = *((char *)r + (_CTRL_OFFSET+_G_OFFSET));
r = *((char *)r + (_KARCH_OFFSET+_TBR_OFFSET));
```

macro GET_CKTHREAD_FLAGS r tmp

```
r = GET_CKTHREAD();
r = *((char *)r + (_CTRL_OFFSET+_FLAGS_OFFSET));
```

macro SET_CKTHREAD_SWTRAP swtrap, t, tmp

```
t = GET_CKTHREAD();
t = *((char *)t + (_CTRL_OFFSET+_G_OFFSET));
*((char *)t + _SWTRAP_OFFSET) = swtrap
```

macro FROM_SV psr, to

```
if (psr & 0x40) {
    // == 1 supervisor mode
    break;
} else {
    goto to;
}
```

macro FROM_USR psr, to

```
if (psr & 0x40) {
    goto to;
} else {
    // != 1 user mode
    break;
}
```

macro SAVE_CWND r

```
//TODO
```

macro RESTORE_CWND r

```
//TODO
```

macro SAVE_REGRW sp, r

```
//TODO
```

macro RESTORE_REGRW sp, r

```
//TODO
```

macro SAVE_IRQCTXT sp

```
//TODO
```

macro RESTORE_IRQCTXT sp

```
//TODO
```

macro CKTHREAD_CTRL_GET_IFLAGS r, iFlags

```
char * tmp;
tmp = GET_CKTHREAD();
tmp = *(tmp + (_CTRL_OFFSET+_G_OFFSET));
tmp = *(tmp + (_PARTCTRLTAB_OFFSET));
iFlags = *(tmp + (_IFLAGS_OFFSET+_VAL_OFFSET));
```

macro CKTHREAD_CTRL_RESTORE_IFLAGS r, iFlags

```
*((char *)r + _IFLAGS_OFFSET+_VAL_OFFSET) = iFlags;
```

macro CKTHREAD_CTRL_SET_IFLAGS r, iFlags, tmp

```
char * t;
t = GET_CKTHREAD();
t = *(t + _CTRL_OFFSET+_G_OFFSET);
t = *(t + _PARTCTRLTAB_OFFSET);
*(t + _IFLAGS_OFFSET+_VAL_OFFSET) = iFlags;
```

macro DO_FLUSH_CACHE r0, r1, r2, r3

```c r0 = GET_CKTHREAD(); r1 = *(r + *CTRL_OFFSET*+FLAGS_OFFSET);

r2 = r1 >> 7; r1 = r1 >> 5; r1 = r1 & r2;

//andcc _r1(), 2, %g0 //bne 61f if (r1 & 2) { //61 FLUSH_ICACHE(r2, r3); } if (r1 & 1) { //63 FLUSH_DCACHE(r2, r3); } //62 r1 = *(r0 + CTRL_OFFSET+FLAGS_OFFSET); r1 = r1 andnot (1<<5 | 1<<6); //if flush this time; clear the bit* (r0 + *CTRL_OFFSET*+FLAGS_OFFSET) = r1; ```
```