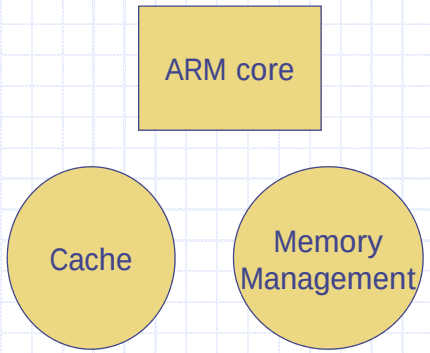
ARM Cache Memory

ARM Processor Organization



Cache Policy

- Write policy: WB or WT
- Cache line replacement policy
  - Round-robin
  - Pseudorandom
  - When victim counter reaches the max. vlaue, reset to the base value
- Allocation policy on a cache miss
  - Read-allocate
  - Read-write-allocate
- E.g., ARM920T: WB/WT, RR/PR, RA
  Xscale: WB/WT, RR, RA/RWA

RR vs. Pseudorandom Replacement

- RR
  - (+) Predictability improved
  - (-) large change in performance for a small change in memory access

EX: ARM 940T Cache

- 4KB I-Cache
- 4 Sets, each set with 1KB
- 16B line size
- 64 B in a way


```
void cache_RRtest(unsigned int times,unsigned int numset)
{
  clock_t count;
  printf("Round Robin test size = %d\r\n", numset);
  enableRoundRobin();
  cleanFlushCache();
  count = clock();
  readSet(times,numset);
  count = clock() - count;
  printf("Round Robin enabled = %.2f seconds\r\n",
            (float)count/CLOCKS_PER_SEC);
  enableRandom();
  cleanFlushCache();
  count = clock();
  readSet(times, numset);
  count = clock() - count;
  printf("Random enabled = %.2f seconds\r\n\r\n",
            (float)count/CLOCKS_PER_SEC);
}
```



```
int readSet( unsigned int times, unsigned int numset)
{
  int setcount, value;
  volatile int *newstart;
  volatile int *start = (int *)0x20000;
  __asm
  {
    timesloop:
      MOV    newstart, start
      MOV    setcount, numset
    setloop:
      LDR    value,[newstart,#0];
      ADD    newstart,newstart,#0x40;
      SUBS   setcount, setcount, #1;
      BNE    setloop;
      SUBS   times, times, #1;
      BNE    timesloop;
  }
  return value;
}
```

Cache_RRtest(0x10000, 64) vs.
Cache_RRtest(0x10000, 65)?

System Control Coprocessor 15

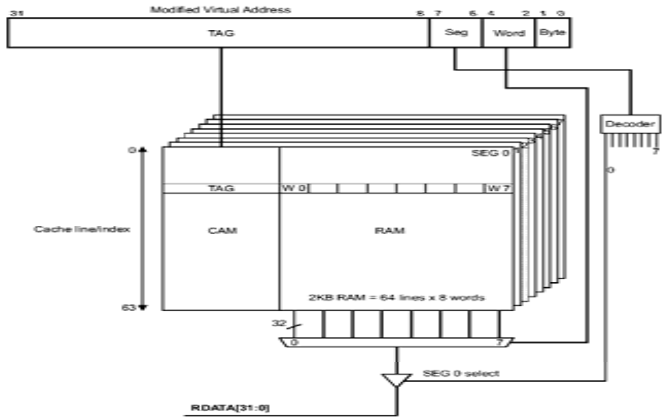
- Used to configure & control ARM cached cores
- Clean & Flush cache
  - Flush in ARM == Invalidate
    - Clear the valid bit in the cache line
  - Clean in ARM == Flush
    - Write dirby cache lines to memory
- Drain write buffer
- Cache lockdown

Cache Lockdown

- Improve the predictability but reduce the cache size
- Candidates for lockdown
  - Vector interrupt table
  - ISR
  - Performance critical code
  - Global variables (frequently used)

ARM 920T I-Cache Organization

- 16KB (512 lines * 32 bytes, arranged as a 64-way set associative cache)



EX: I-Cache Locking

```
        ADRL    r0,start_address        ; address pointer
        ADRL    r1,end_address
        MOV     r2,#lockdown_base<<26   ; victim pointer
        MCR     p15,0,r2,c9,c0,1        ; write ICache victim and lockdown base

loop    MCR     p15,0,r0,c7,c13,1       ; Prefetch ICache line
        ADD     r0,r0,#32               ; increment address pointer to next ICache line

;; do we need to increment the victim pointer?
;; test for segment 0, and if so, increment the victim pointer
;; and write the ICache victim and lockdown base.
        AND     r3,r0,#0xE0             ; extract the segment bits from the address
        CMP     r3,#0x0                 ; test for segment 0
        ADDEQ   r2,r2,#0x1<<26          ; if segment 0, increment victim pointer
        MCREQ   p15,0,r2,c9,c0,1        ; and write ICache victim and lockdown base

;; have we linefilled enough code?
;; test for the address pointer being less than or equal to the
;; end_address and if so, loop and perform another linefill
        CMP     r0,r1                   ; test for less than or equal to end_address
        BLE     loop                    ; if not, loop

;; have we exited with r3 pointing to segment 0?
;; if so, the ICache victim and lockdown base has already been set to one
;; higher than the last entry written.
;; if not, increment the victim pointer and write the ICache victim and
;; lockdown base.
        CMP     r3,#0x0                 ; test for segments 1 to 7
        ADDNE   r2,r2,#0x1<<26          ; if address is segment 1 to 7,
        MCRNE   p15,0,r2,c9,c0,1        ; write ICache victim and lockdown base
```