AMD

# Implementation of Write Allocate in the K86™ Processors

*Application Note*

Publication # **21326** Rev: **F** Amendment/**0**
Issue Date: **February 1999**

# Contents

# List of Figures

# Revision History

| Date | Rev | Description |
|---|---|---|
| Sept 1997 | C | Modified description of WCDE bit in Write Handling Control Register (WHCR) Model-Specific Register. See pages 4 and 8. |
| May 1998 | D | Switched order of the AMD-K6 and AMD-K5 information. |
| May 1998 | D | Revised “Step 1: Determine Processor Model and Stepping” on page 3. |
| May 1998 | D | Revised “Step 2: AMD-K6® Processor Models 6, 7 and AMD-K6-2 Processor Model 8/[7:0]” on page 3. |
| Nov 1998 | E | Added “Step 2: AMD-K6®-2 Processor Model 8/[F:8] and AMD-K6-III Processor Model 9” on page 6. |
| Nov 1998 | E | Added case 3 and code sample to “AMD-K6® Processor Programming Example for Write Allocate Registers” on page 8. |
| Feb 1999 | F | Added ”Models 6, 7, and 8” and “Model 9” on page 1 and “All AMD-K6® Models” on page 2. |
| Feb 1999 | F | Revised ”Write Allocate Enable Limit” and ”Write Allocate Enable 15-to-16-Mbyte” paragraphs in both ”Step 2: AMD-K6® Processor Models 6, 7 and AMD-K6-2 Processor Model 8/[7:0]” and ”Step 2: AMD-K6®-2 Processor Model 8/[F:8] and AMD-K6-III Processor Model 9”. |
| Feb 1999 | F | Added AMD-K6-III information to “Step 2: AMD-K6®-2 Processor Model 8/[F:8] and AMD-K6-III Processor Model 9” on page 6. |

# *Application Note*

# Implementation of Write Allocate in the K86™ Processors

## What is Write Allocate?

Write allocate, if enabled, occurs when the processor has a pending memory write cycle to a cacheable line and the line does not currently reside in the Level-1 (L1) data cache. The behavior of the processor thereafter differs depending on whether the processor model contains a Level-2 (L2) cache. The AMD-K6® processor Model 6 and Model 7, and the AMD-K6-2 processor Model 8 do not contain a L2 cache, whereas the AMD-K6-III processor Model 9 does contain a L2 cache.

**Models 6, 7, and 8**

After the L1 data cache miss, the processor performs a 32-byte burst read cycle on the system bus to fetch the data-cache line addressed by the pending write cycle. The data associated with the pending write cycle is merged with the recently-allocated cache line and stored in the processor's L1 data cache. The final MESI (Modified, Exclusive, Shared, Invalid) state of the cache line depends on the state of the WB/WT# and PWT signals during the burst read cycle and the subsequent cache write hit.

**Model 9**

After the L1 data cache miss, the processor checks for the cache line in the L2 cache. If the line does not exist in the L2 cache, the processor performs a 32-byte burst read cycle on the system bus to fetch the data-cache line addressed by the pending write cycle. If the line does exist in the L2 cache, the data is supplied directly from the L2 cache, in which case a system bus cycle is

not executed. The data associated with the pending write cycle is merged with the recently-allocated data-cache line and stored in the processor's L1 data cache. If the data-cache line was fetched from memory (because of an L2 cache miss), the data is stored without modification in the L2 cache. The final MESI state of the cache lines depends on the state of the WB/WT# and PWT signals during the burst read cycle and the subsequent L1 data cache write hit. If the L1 data cache line is stored in the modified state, then the same cache line is stored in the L2 cache in the exclusive state. If the L1 data cache line is stored in the shared state, then the same cache line is stored in the L2 cache in the shared state.

**All AMD-K6® Models**

During the write allocation, a 32-byte burst read cycle is executed in place of a non-burst write cycle (in the case of the Model 9, this assumes the data-cache line was not present in the L2 cache). While the burst read cycle generally takes longer to execute than the non-burst write cycle, performance gains are realized on subsequent write cycle hits to the write-allocated cache line. Due to the nature of software, memory accesses tend to occur within proximity of each other (principle of locality). The likelihood of additional write hits to the write-allocated cache line is high.

For the Model 9, write allocates that hit the L2 cache increase performance by avoiding accesses to the system bus.

# Programming Details

The steps required for programming write allocate on K86™ processors are as follows:

1. Verify write allocate support by using the CPUID instruction to check for the correct model and stepping of the processor.
2. Configure the Model-Specific Registers (MSRs).
3. Enable write allocate.

   ***Note:*** *The BIOS should enable the write allocate mechanisms only after performing any memory sizing or typing algorithms.*

## Step 1: Determine Processor Model and Stepping

The first step in supporting the write allocate feature of the AMD K86 processors is determining the model and stepping of the processor.

**Write Allocate Support**

Write allocate is supported on every stepping for every model of the AMD-K6 processor. Write allocate in the AMD-K5™ processor is supported only on the following models with a stepping of 4 or greater: Models 1, 2, and 3. Use the CPUID instruction to determine if the proper model and stepping of the processor is present. See the *AMD Processor Recognition Application Note*, order# 20734 for more information.

**MSR Format**

After determining that the processor supports write allocate, the next step is to configure the corresponding MSR that enables write allocate. This MSR on the AMD-K6 processor is the Write Handling Control Register (WHCR), which has two formats. AMD-K6 processor Models 6 and 7 and AMD-K6-2 processor Model 8 steppings 0 through 7 use the same WHCR format. AMD-K6-2 processor Model 8 steppings 8 through F and AMD-K6-III processor Model 9 use a different WHCR format.

For AMD-K6 processors, go to either "Step 2: AMD-K6® Processor Models 6, 7 and AMD-K6-2 Processor Model 8/[7:0]" on page 3 or "Step 2: AMD-K6®-2 Processor Model 8/[F:8] and AMD-K6-III Processor Model 9" on page 6.

For an AMD-K5 processor Models 1, 2, or 3 with a stepping of 4 or greater, go to "Step 2: AMD-K5™ Processor" on page 10.

## Step 2: AMD-K6® Processor Models 6, 7 and AMD-K6-2 Processor Model 8/[7:0]

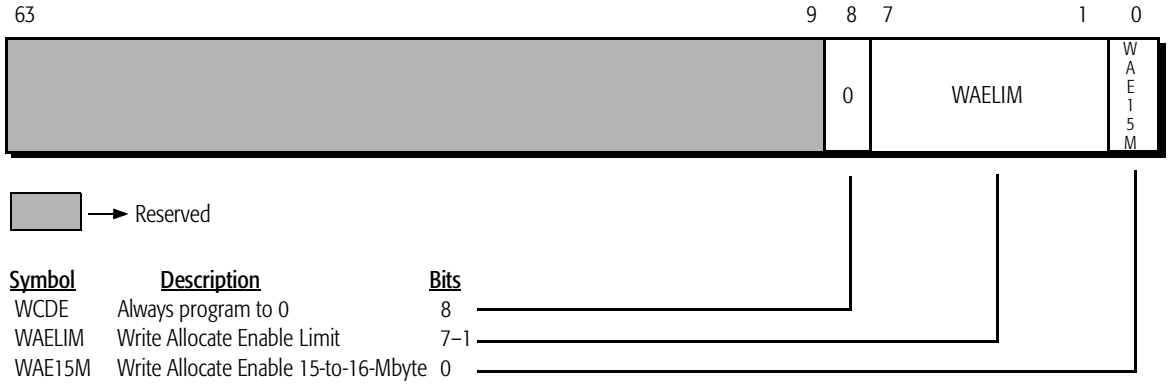
The AMD-K6 processor uses two mechanisms (programmable within the WHCR) to determine when to perform write allocates. A write allocate is performed when either of these mechanisms detects that a pending write is to a cacheable area of memory.

Before programming the WHCR or changing memory cacheability/writeability, the BIOS must writeback and invalidate the internal cache by using the WBINVD instruction. In addition, the WHCR should enable the write allocate

mechanisms only after performing any memory sizing or typing algorithms.

### Write Handling Control Register (WHCR)

The WHCR contains three fields—the WCDE bit, the Write Allocate Enable Limit (WAELIM) field, and the Write Allocate Enable 15-to-16-Mbyte (WAE15M) bit (See Figure 1).




| Symbol | Description | Bits |
|---|---|---|
| WCDE | Always program to 0 | 8 |
| WAELIM | Write Allocate Enable Limit | 7–1 |
| WAE15M | Write Allocate Enable 15-to-16-Mbyte | 0 |

***Note**: Hardware RESET initializes this MSR to all zeros.*

**Figure 1. Write Handling Control Register (WHCR)—MSR C000_0082h (Models 6, 7, and 8/[7:0])**

**WCDE.** For proper functionality, always program bit 8 of WHCR to 0.

**Write Allocate Enable Limit.** The WAELIM field is 7 bits wide. This field, multiplied by 4 Mbytes, defines an upper memory limit. Any pending write cycle that misses the L1 cache and that addresses memory below this limit causes the processor to perform a write allocate (assuming the address is not within a range where write allocates are disallowed). Write allocate is disabled for memory accesses at and above this limit unless the processor determines a pending write cycle is cacheable by means of one of the other write allocate mechanisms—"Write to a Cacheable Page" and "Write to a Sector" (for more information, see the "Cache Organization" chapter in the *AMD-K6®Processor Data Sheet*, order# 20695 or the *AMD-K6®-2 Processor Data Sheet*, order# 21850). The maximum value of this limit is $((2^7-1) \cdot 4$ Mbytes) = 508 Mbytes. When all the bits in this field are set to 0, all memory is above this limit and the write allocate mechanism is disabled (even if all bits in the WAELIM field are set to 0, write allocates can still occur due to the "Write to a Cacheable Page" and "Write to a Sector" mechanisms).

Once the BIOS determines the amount of RAM installed in the system, this number should also be used to program the WAELIM field. For example, a system with 32 Mbytes of RAM would program the WAELIM field with the value 000_1000b. This value (8), when multiplied by 4 Mbytes, yields 32 Mbytes as the write allocate limit.

**Write Allocate Enable 15-to-16-Mbyte.** The WAE15M bit is used to enable write allocations for memory write cycles that address the 1 Mbyte of memory between 15 Mbytes and 16 Mbytes. This bit must be set to 1 to allow write allocates in this memory area. This sub-mechanism of the WAELIM provides a memory hole to prevent write allocates. This memory hole is provided to account for a small number of uncommon memory-mapped I/O adapters that use this particular memory address space. If the system contains one of these peripherals, the bit should be set to 0 (even if the WAE15M bit is set to 0, write allocates can still occur between 15 Mbytes and 16 Mbytes due to the "Write to a Cacheable Page" and "Write to a Sector" mechanisms). The WAE15M bit is ignored if the value in the WAELIM field is set to less than 16 Mbytes.

By definition, write allocations in the AMD-K6 are not performed in the memory area between 640 Kbytes and 1 Mbyte unless the processor determines a pending write cycle is cacheable by means of "Write to a Cacheable Page" or "Write to a Sector." It is not safe to perform write allocations between 640 Kbytes and 1 Mbyte (000A_0000h to 000F_FFFFh) because it is considered a noncacheable region of memory.

To complete programming write allocate on the AMD-K6 processor, go to "Step 3: AMD-K6® Processor (All Models)" on page 8.

## Step 2: AMD-K6®-2 Processor Model 8/[F:8] and AMD-K6-III Processor Model 9

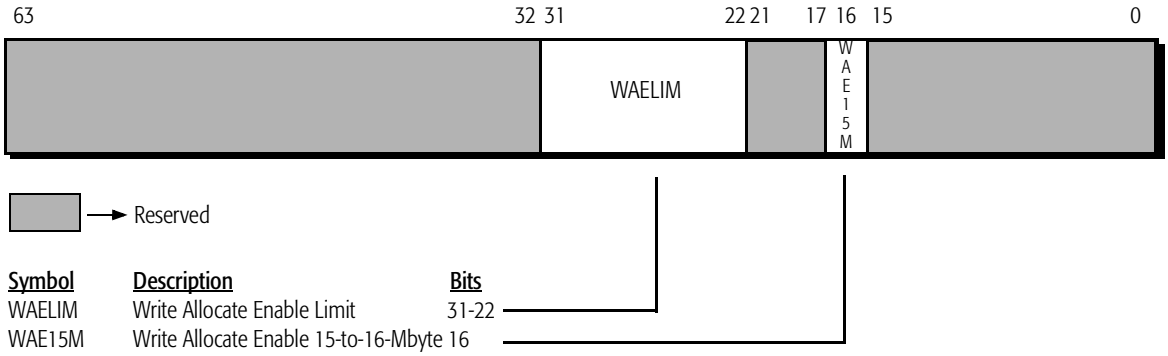
The AMD-K6-2 processor and the AMD-K6-III processor use two mechanisms (programmable within the WHCR) to determine when to perform write allocates. A write allocate is performed when either of these mechanisms detects that a pending write is to a cacheable area of memory.

Before programming the WHCR or changing memory cacheability/writeability, the BIOS must writeback and invalidate the internal cache by using the WBINVD instruction. In addition, the WHCR should enable the write allocate mechanisms only after performing any memory sizing or typing algorithms.

### Write Handling Control Register (WHCR)

The WHCR contains two fields—the Write Allocate Enable Limit (WAELIM) field, and the Write Allocate Enable 15-to-16-Mbyte (WAE15M) bit (see Figure 2).

***Note:*** *The WHCR register as defined in the Model 6, Model 7, and Model 8/[7:0] has changed in the Model 8/[F:8] and Model 9.*




***Note****: Hardware RESET initializes this MSR to all zeros.*

**Figure 2. Write Handling Control Register (WHCR)—MSR C000_0082h (Model 8/[F:8] and Model 9)**

**Write Allocate Enable Limit.** The WAELIM field is 10 bits wide. This field, multiplied by 4 Mbytes, defines an upper memory limit. Any pending write cycle that misses the L1 cache and that addresses memory below this limit causes the processor to perform a write allocate (assuming the address is not within a range where write allocates are disallowed). Write allocate is disabled for memory accesses at and above this limit unless the processor determines a pending write cycle is cacheable by

means of one of the other write allocate mechanisms—"Write to a Cacheable Page" and "Write to a Sector" (for more information, see the "Cache Organization" chapter in the *AMD-K6®-2 Processor Data Sheet*, order# 21850 or *AMD-K6®-III Processor Data Sheet*, order# 21918). The maximum value of this limit is $((2^{10}-1) \cdot 4$ Mbytes) = 4092 Mbytes. When all the bits in this field are set to 0, all memory is above this limit and the write allocate mechanism is disabled (even if all bits in the WAELIM field are set to 0, write allocates can still occur due to the "Write to a Cacheable Page" and "Write to a Sector" mechanisms).

Once the BIOS determines the amount of RAM installed in the system, this number should also be used to program the WAELIM field. For example, a system with 32 Mbytes of RAM would program the WAELIM field with the value 00_0000_1000b. This value (8), when multiplied by 4 Mbytes, yields 32 Mbytes as the write allocate limit.

**Write Allocate Enable 15-to-16-Mbyte.** The WAE15M bit is used to enable write allocations for memory write cycles that address the 1 Mbyte of memory between 15 Mbytes and 16 Mbytes. This bit must be set to 1 to allow write allocates in this memory area. This sub-mechanism of the WAELIM provides a memory hole to prevent write allocates. This memory hole is provided to account for a small number of uncommon memory-mapped I/O adapters that use this particular memory address space. If the system contains one of these peripherals, the bit should be set to 0 (even if the WAE15M bit is set to 0, write allocates can still occur between 15 Mbytes and 16 Mbytes due to the "Write to a Cacheable Page" and "Write to a Sector" mechanisms). The WAE15M bit is ignored if the value in the WAELIM field is set to less than 16 Mbytes.

By definition, write allocations are not performed in the memory area between 640 Kbytes and 1 Mbyte unless the processor determines a pending write cycle is cacheable by means of "Write to a Cacheable Page" or "Write to a Sector." It is not safe to perform write allocations between 640 Kbytes and 1 Mbyte (000A_0000h to 000F_FFFFh) because it is considered a noncacheable region of memory. Additionally, if a memory region is defined as write-combinable or uncacheable by a Memory Type Range Register (MTRR), write allocates are not performed in that region.

## Step 3: AMD-K6® Processor (All Models)

The BIOS programmer has several options regarding what the end-user can control. The BIOS can provide the end-user with a setup screen option to enable/disable write allocate or options to define the Write Allocate Enable Limit field and set the Write Allocate Enable 15-to-16-Mbyte bit. The BIOS can also automatically enable and setup the write allocate feature and its registers without end-user intervention. This automatic setup is recommended. To disable all write allocate features for the AMD-K6 processor, the WHCR must be set to 0000_0000_0000_0000h—the default value after power-on reset.

## AMD-K6® Processor Programming Example for Write Allocate Registers

The following cases show examples of programming the write allocate feature for three types of systems:

### Case 1

For AMD-K6 processor Models 6 and 7 and AMD-K6-2 processor Model 8/[7:0] systems that have a 1-Mbyte memory hole starting at the 15-Mbyte boundary, and 32 Mbytes of total memory:

- Program the WHCR MSR (ECX=C000_0082h) with WCDE=0, WAELIM=8, and WAE15M=0
  - Use the WRMSR instruction and the 64-bit hex value 0000_0000_0000_0010h

### Code Sample

```
;flush cache
PUSHF                       ;save state
CLI                         ;disable interrupts
WBINVD                      ;write back and invalidate cache
;set Write Allocate Limit and clear WAE15M bit
MOV      ECX,0C0000082H
MOV      EAX,10H            ;WCDE=0,WAELIM=8,WAE15M=0
XOR      EDX,EDX
WRMSR
POPF                        ;restore original state
```

**Case 2**

For AMD-K6 processor Models 6 and 7 and AMD-K6-2 processor Model 8/[7:0] systems that do not have a memory hole starting at the 15-Mbyte boundary, and have 16 Mbytes of total memory:

- Program the WHCR MSR (ECX=C000_0082h) with WCDE=0, WAELIM=4, and WAE15M=1
  - Use the WRMSR instruction and the 64-bit hex value 0000_0000_0000_0009h

**Code Sample**

```
;flush cache
PUSHF                       ;save state
CLI                         ;disable interrupts
WBINVD                      ;write back and invalidate cache
;set Write Allocate Limit and set WAE15M bit
MOV      ECX,0C0000082H
MOV      EAX,09H            ;WCDE=0,WAELIM=4,WAE15M=1
XOR      EDX,EDX
WRMSR
POPF                        ;restore original state
```

**Case 3**

For AMD-K6-2 processor Model 8/[F:8] and AMD-K6-III processor Model 9 systems that do not have a memory hole starting at the 15-Mbyte boundary, and have 64 Mbytes of total memory:

- Program the WHCR MSR (ECX=C000_0082h) with WAELIM=16d and WAE15M=1
  - Use the WRMSR instruction and the 64-bit hex value 0000_0000_0401_0000h

**Code Sample**

```
;flush cache
PUSHF                       ;save state
CLI                         ;disable interrupts
WBINVD                      ;write back and invalidate cache
;set Write Allocate Limit and set WAE15M bit
MOV      ECX,0C0000082H
MOV      EAX,04010000H      ;WAELIM=16d,WAE15M=1
XOR      EDX,EDX
WRMSR
POPF                        ;restore original state
```

## Step 2: AMD-K5™ Processor

The AMD-K5 processor implements write allocate by providing a global write allocate enable bit, three range-protection enable bits, and two memory range registers. The global write allocate enable bit is accessed using the Hardware Configuration Register (HWCR). The memory range registers and range enable bits are programmed by read/write MSR instructions.

The Write Allocate Enable bit (bit 4 of HWCR) should be set to 0, which prevents potential erroneous behavior in the case of a warm boot during write allocate initialization.

Two MSRs are defined to support write allocate. The MSRs are accessed using the RDMSR and WRMSR instructions (see "RDMSR and WRMSR" in the *AMD-K5™ Processor Software Development Guide*, order# 20007). The following index values in the ECX register access the MSRs:

- Write Allocate Top-of-Memory and Control Register (WATMCR)—ECX = 85h
- Write Allocate Programmable Memory Range Register (WAPMRR)—ECX = 86h

Three non-write-allocatable memory ranges are defined for use with the write allocate feature—one fixed range and two programmable ranges.

**Fixed Range.** The fixed memory range is 000A_0000h–000F_FFFFh and can be enabled or disabled. When enabled, write allocate can not be performed in this range.

This region of memory, which includes standard VGA and other peripheral and BIOS access, is considered noncacheable. Performing a write allocate in this area can cause compatibility problems. It is recommended that this bit be enabled (set to 1) to prevent write allocate to this range. Set bit 16 of WATMCR to enable protection of this range.

**Programmable Range.** One programmable memory range is xxxx_0000h–yyyy_FFFFh, where xxxx and yyyy are defined using bits 15–0 and bits 31–16 of WAPMRR, respectively. Set bit 17 of WATMCR to enable protection of this range. When enabled, write allocate can not be performed in this range.

This programmable memory range exists because a small number of uncommon memory-mapped I/O adapters are mapped to physical RAM locations. If a card like this exists in the system configuration, it is recommended that the BIOS program the 'memory hole' for the adapter into this non-write-allocatable range.

**Top of Memory.** The other programmable memory range is defined by the 'top-of-memory' field. The top of memory is equal to zzzz_0000h, where zzzz is defined using bits 15–0 of WATMCR. Addresses above zzzz_0000h are protected from write allocate when bit 18 of WATMCR is enabled.

Once the BIOS determines the size of RAM installed in the system, this size should also be used to program the top of memory. For example, a system with 32 Mbytes of RAM requires that the top-of-memory field be programmed with a value of 0200h, which enables protection from write allocate for memory above that value. Set bit 18 of WATMCR to enable protection of this range.

Caching and write allocate are generally not performed for the memory above the amount of physical RAM in the system. Video frame buffers are usually mapped above physical RAM. If write allocate were attempted in that memory area, there could be performance degradation or compatibility problems.

Bits 18–16 of WATMCR control the enabling or disabling of the three memory ranges as follows:

- Bit 18: Top-of-Memory Enable bit

  0 = disabled (default)
  1 = enabled (write allocate can not be performed above Top of Memory)
- Bit 17: Programmable Range Enable bit

  0 = disabled (default)

  1 = enabled (write allocate can not be performed in this range)
- Bit 16: Fixed Range Enable bit

  0 = disabled (default)

  1 = enabled (write allocate can not be performed in this range)

Figures 3 and 4 show the bit positions for these two new registers.

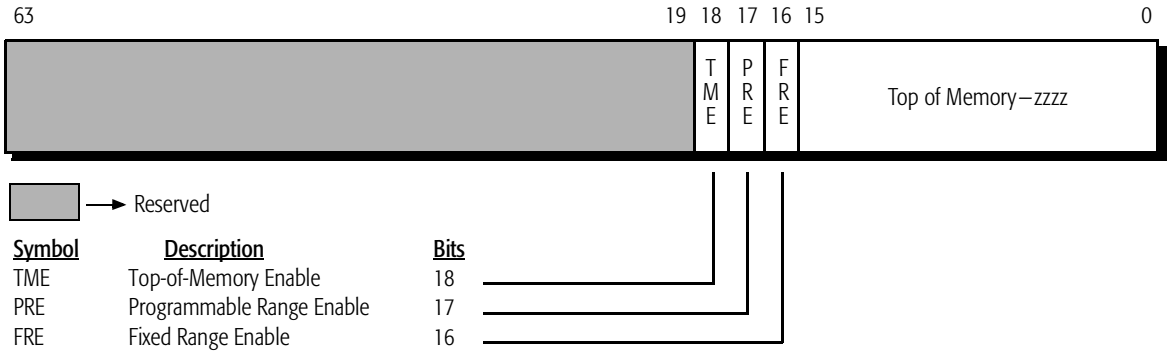



**Figure 3. Write Allocate Top-of-Memory and Control Register (WATMCR)—MSR 85h**

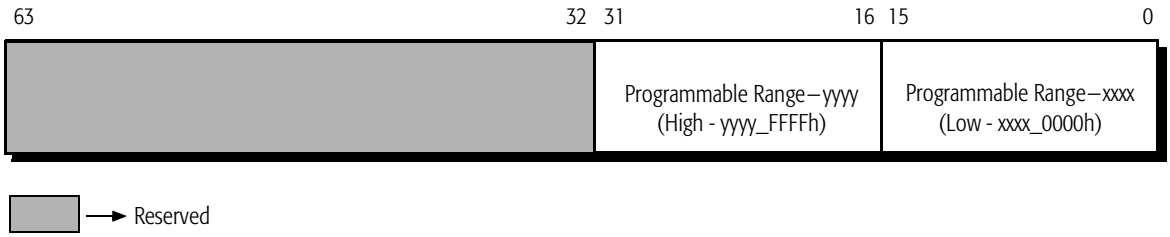



**Figure 4. Write Allocate Programmable Memory Range Register (WAPMRR)—MSR 86h**

## Step 3: AMD-K5™ Processor

All of the write allocate features in the AMD-K5 processor are enabled by setting bit 4 (WA) of the HWCR (MSR 83h) to 1. For more information on the HWCR, see "Hardware Configuration Register" in the *AMD-K5™ Processor Software Development Guide*, order# 20007. Figure 5 shows the definition of HWCR.

The BIOS programmer has several options regarding what the end-user can control. The BIOS can provide the end-user with a setup screen option to enable write allocate. The BIOS can provide the end-user with a setup screen option to also setup the other features (programmable ranges and fixed range). The BIOS can automatically enable and setup the write allocate

feature and its registers without end-user intervention. This automatic setup is recommended.

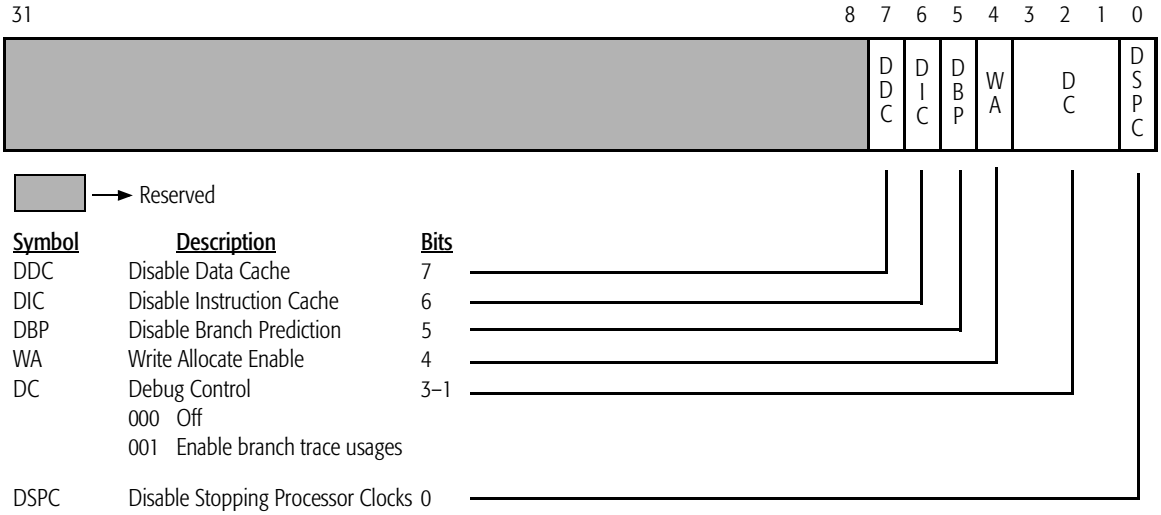



**Figure 5. Hardware Configuration Register (HWCR)—MSR 83h**

## AMD-K5™ Processor Programming Example for Write Allocate Registers

The following cases show examples of programming the write allocate feature for two types of systems:

**Case 1**

For systems without a memory hole and 16 Mbytes of total memory:

- Program the WATMCR MSR (ECX=85h) with top of memory (0100h) and enable bits (0005h) to protect the fixed range and above the top of memory
  - Use the WRMSR instruction and the 64-bit hex value 0000_0000_0005_0100h

***Note:*** *For 8-Mbyte systems, program 0080h in the lowest 16 bits. For 32-Mbyte systems, program 0200h in the lowest 16 bits.*

**Code Sample**

```
;disable WA bit (bit 4 of HWCR)
MOV      ECX,83H           ;read HWCR (83h)
RDMSR
AND      EAX,NOT 10H
WRMSR
;program top-of-memory and control bits
MOV      ECX,85H           ;select WATMCR
MOV      EAX,50100H        ;TME=1,PRE=0,FRE=1,TOM=0100h
XOR      EDX,EDX
WRMSR
```

```
;enable WA bit
MOV      ECX,83H           ;read HWCR (83h)
RDMSR
OR       EAX,10H           ;set bit 4
WRMSR
```

**Case 2**

For systems with a 1-Mbyte memory hole starting at the 15 Mbyte boundary and 32 Mbytes of total memory:

- Program the WAPMRR MSR (ECX=86h) with 15 Mbytes (00F0h) to 16 Mbytes –1 (00FFh)
  - Use the WRMSR instruction and the 64-bit hex value 0000_0000_00FF_00F0h
- Program the WATMCR MSR (ECX=85h) with top of memory (0200h) and all enable bits (0007h) to protect above the top of memory, the fixed range, and the programmable range
  - Use the WRMSR instruction and the 64-bit hex value 0000_0000_0007_0200h

***Note:*** *For 8-Mbyte systems, program 0080h in the lowest 16 bits. For 16-Mbyte systems, program 0100h in the lowest 16 bits.*

**Code Sample**

```
;disable WA bit (bit 4 of HWCR)
MOV      ECX,83H           ;read HWCR (83h)
RDMSR
AND      EAX,NOT 10H
WRMSR
;program programmable range to 15-16Mbytes
MOV      ECX,86H           ;select WAPMRR
MOV      EAX,0FF00F0H      ;address from F00000 to FFFFFF
XOR      EDX,EDX           ;clear
WRMSR
;program top of memory and control bits
MOV      ECX,85H           ;select WATMCR
MOV      EAX,70200H        ;TME=1,PRE=1,FRE=1,TOM=0200h
XOR      EDX,EDX           ;clear
WRMSR
;enable WA bit
MOV      ECX,83H           ;read HWCR (83h)
RDMSR
OR       EAX,10H           ;set bit 4
WRMSR
```