Vircon32

CONSOLA VIRTUAL DE 32 BITS

Especificación del sistema

Parte 9: Formatos de archivo

Documento con fecha 2024.01.14 Escrito por Carra

¿Qué es esto?

Este documento es la parte número 9 de la especificación del sistema Vircon32. Esta serie de documentos define el sistema Vircon32, y provee una especificación completa que describe en detalle sus características y comportamiento.

El principal objetivo de esta especificación es definir un estándar de lo que es un sistema Vircon32, y cómo debe implementarse un sistema de juego para que se considere conforme a él. Además, al ser Vircon32 es un sistema virtual, un importante objetivo adicional de estos documentos es proporcionar a cualquiera el conocimiento para crear sus propias implementaciones de Vircon32.

Sobre Vircon32

El proyecto Vircon32 fue creado de forma independiente por Carra. El sistema Vircon32 y su material asociado (incluyendo documentos, software, código fuente, arte y cualquier otro elemento relacionado) son propiedad del autor original.

Vircon32 es un proyecto libre y de código abierto en un esfuerzo por promover que cualquiera pueda jugar a la consola y crear software para ella. Para obtener información más detallada al respecto, se recomienda consultar los textos de licencia incluidos en cada uno de los programas disponibles.

Sobre este documento

Índice

La parte 9 de la especificación describe los formatos de archivo usados para toda la información que una implementación de Vircon32 necesita cargar y/o guardar durante su funcionamiento. Esto incluye cabeceras y estructura de archivos, formatos de datos y validaciones básicas de la corrección de los archivos.

1 Introducción

Para su funcionamiento, un sistema Vircon32 necesita poder trabajar con información externa. En concreto necesitará leer el contenido de BIOS y cartuchos, y leer y escribir el contenido de tarjetas de memoria. Consideraremos que estos contenidos externos se representan como "archivos".

Cada implementación puede manejarlos con diferentes sistemas de archivos y guardarlos en distintos soportes físicos. Por eso abstraemos esos detalles: para nuestros fines, un "archivo" es simplemente una secuencia ordenada de bytes. Opcionalmente puede tener también un nombre asociado. También consideramos que un archivo es persistente: su contenido seguirá almacenado y accesible aún si se apaga un sistema Vircon32 concreto.

Todos los archivos que almacenan contenidos externos para sistemas Vircon32 (BIOS, cartuchos y tarjetas de memoria) deben ser intercambiables entre diferentes consolas Vircon32. Para asegurar la compatibilidad entre distintas implementaciones, este documento describe los formatos que cada archivo usa para representar su información.

Obsérvese que los formatos de representación descritos en este documento sólo son obligatorios para el intercambio de información (es decir, información que cruzará la frontera de un sistema Vircon32). Sin embargo, las implementaciones pueden usar otros formatos para representaciones o procesos internos. Por ejemplo, una implementación hardware puede preferir una representación interna diferente para las texturas de la GPU si esto permite un acceso más sencillo a los pixels desde el hardware de vídeo.

2 Tipos de archivos en Vircon32

En primer lugar enumeramos los distintos tipos de archivos que usan los sistemas Vircon32 y sus características básicas. Cada uno de estos tipos de archivo se describirá en detalle en las siguientes secciones.

Archivos ROM

Las ROMs son los archivos principales que usa cualquier sistema Vircon32. Una ROM es un archivo ejecutable que empaqueta un programa junto con las texturas y sonidos que necesita, todo en un único archivo. Vircon32 usa la extensión *.v32 para archivos ROM.

Tanto las BIOS como los cartuchos son ejecutables, por lo que ambos se guardan con este mismo formato. La única diferencia es que, para una BIOS, la firma del archivo cambia y se aplican algunas restricciones adicionales. Esto se detalla en la sección 7.

Archivos de recursos

Como se ha dicho las ROMs reúnen 3 tipos de recursos para un ejecutable: el binario del programa, las texturas de GPU y los sonidos de SPU. Cada recurso también se puede representar como un archivo independiente, dando lugar a estos 3 formatos de archivo:

- **Archivos binarios de programa** (extensión *.vbin)
  Contienen un binario completo (programa + datos) para un único programa.

- **Archivos de textura GPU** (extensión *.vtex)
  Contienen una única textura utilizable por el chip gráfico (GPU).

- **Archivos de sonido SPU** (extensión *.vsnd)
  Contienen un único sonido utilizable por el chip de sonido (SPU).

Debe señalarse que estos recursos sólo están pensados para manejarse como archivos independientes por herramientas de desarrollo u otros programas de soporte. Un sistema Vircon32 por sí mismo normalmente no usará los recursos de esta forma. Aún así, es necesario describirlos porque estos formatos forman parte de los propios archivos ROM.

Un dato importante sobre estos formatos es que los recursos siempre se almacenan sin comprimir. Esto da lugar a archivos más grandes, pero a cambio la lectura/escritura de los archivos es mucho más sencilla. También se evitan dependencias de librerías adicionales de audio y vídeo, al no haber formatos específicos de imagen o sonido.

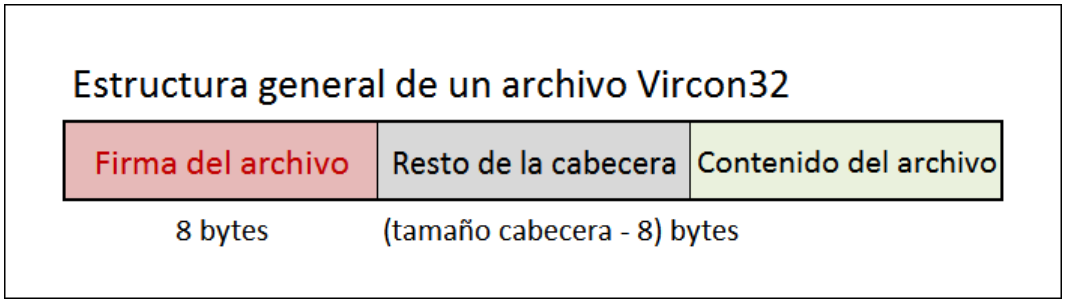
Archivos de tarjeta de memoria

Las tarjetas de memoria son esencialmente regiones de memoria en las que pueden leer y escribir los programas de Vircon32. Al terminar la ejecución, el contenido de esa memoria queda guardado en un archivo. Vircon32 usa la extensión de archivo *.memc para los archivos de tarjetas de memoria.

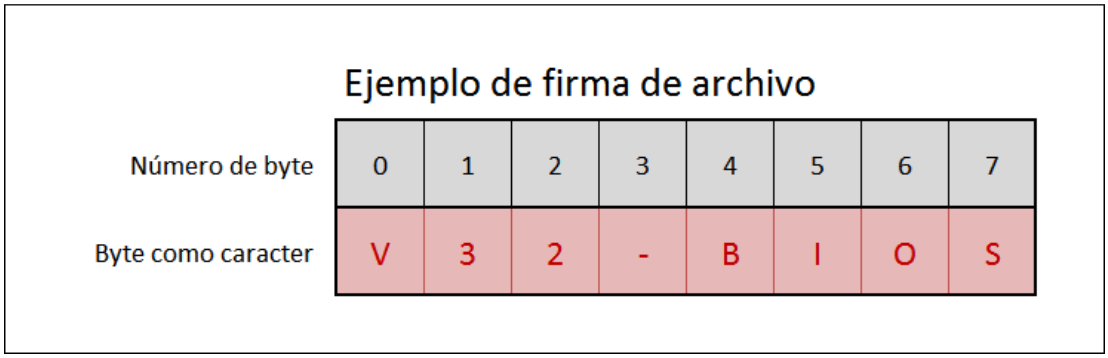
2.1 Firmas de archivo

Usar las extensiones para distinguir entre tipos de archivo es cómodo, pero no es lo bastante fiable. Los usuarios pueden cambiar las extensiones y algunos sistemas de archivos no admiten extensiones. Por ello, el tipo de archivo se debe poder identificar a partir de su contenido.

Para ello, todos los archivos Vircon32 incluyen una "firma de archivo" conocida en sus primeros 8 bytes, que identifica el formato de su contenido. Las siguientes secciones muestran las distintas firmas de archivo como parte de sus cabeceras. La firma de archivo se muestra siempre en rojo, como primer campo en todas las cabeceras de archivo.



Las firmas de archivos se almacenan como una secuencia de 8 caracteres codificados en ASCII estándar. Aún así, como reconocer una firma requiere coincidencia exacta, basta con comparar valores de 64 bits sin ningún formato. Por ejemplo, en un archivo válido de BIOS, esperaremos que los primeros 8 bytes del archivo coincidan con esto:



Debido al uso de firmas, cualquier archivo cuyo tamaño no supere los 8 bytes se puede descartar inmediatamente como archivo no válido para Vircon32.

3 Formatos de datos usados

Todos estos archivos organizan sus datos en campos o secuencias. Cada elemento de un campo o secuencia se codificará utilizando uno de los siguientes formatos de datos. Téngase en cuenta que para todos los formatos de datos de 4 bytes, los bytes están ordenados en sistema little-endian.

3.1 Carácter

Un carácter es un único byte codificado en ASCII. Estos formatos de archivo siempre agrupan los caracteres en arrays de capacidad fija para formar cadenas. Los caracteres se usan en dos tipos distintos de cadenas:

- **Firmas de archivo**. No tienen terminación nula porque una firma siempre tiene exactamente 8 caracteres. Los caracteres utilizados para las firmas de archivo válidas se limitan a ASCII estándar: valores numéricos de 0 a 127.

- **Títulos de ROM**. Como su longitud es variable, usan una terminación nula: se coloca un byte con valor cero después del último carácter usado para marcar dónde termina la cadena. Para los títulos de ROM se permite usar ASCII extendido, que se interpretará según la página de código 1252, también conocida como Latin-1.

3.2 Entero

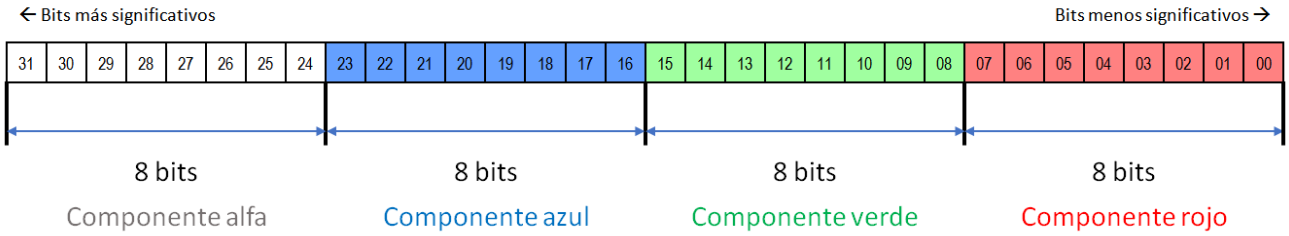
Estos archivos siempre representan enteros con 4 bytes. Cada uno de estos enteros se interpreta como un entero sin signo de 32 bits. Esto equivale al tipo de datos C “uint32_t”.

3.3 Palabra de CPU

Los archivos que contienen datos binarios de programa representan cada una de las palabras que almacenan como un único valor binario de 32 bits. Durante la ejecución del programa, la CPU de Vircon32 puede interpretar cada palabra como una instrucción u otros formatos de datos pero, en el contexto de estos archivos, las palabras almacenadas no requieren más interpretación que una secuencia de 32 bits.

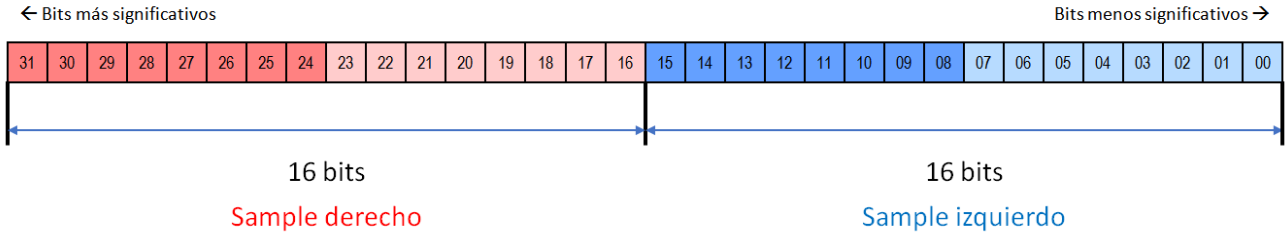
3.4 Pixel de GPU

Los archivos que contienen texturas de GPU representan cada uno de los pixels que almacenan como un único color RGBA de 32 bits, como ya se documentó en la parte 2 de la especificación:



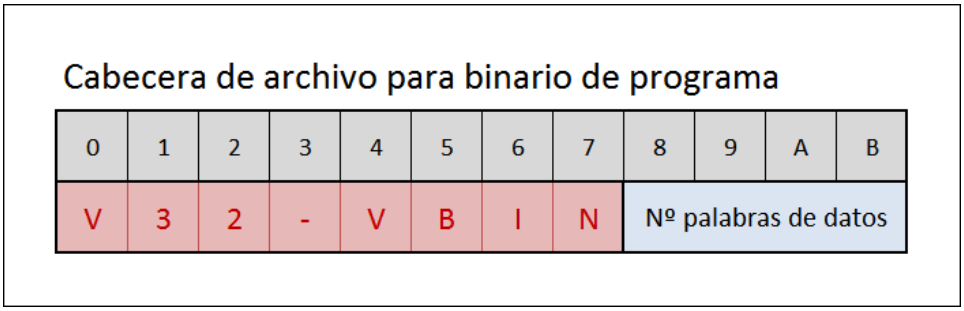
3.5 Sample de SPU

Los archivos que contienen sonidos de SPU representan cada uno de los samples que almacenan como un único valor de 32 bits que agrupa 2 valores de 16 bits para los canales izquierdo y derecho, como ya se documentó en la parte 2 de la especificación:

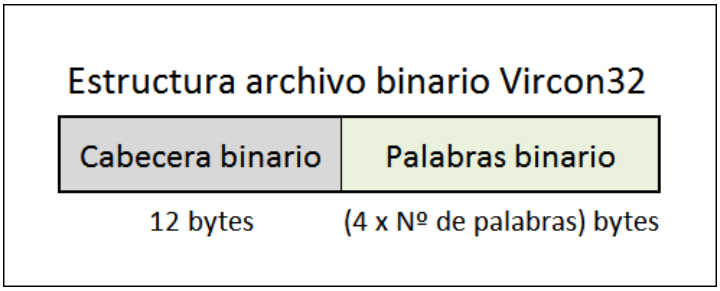


4 Archivos binarios de programa

Aparte de la firma, la cabecera de un archivo binario de programa sólo contiene 1 campo entero que indica el número de palabras de CPU de 32 bits que componen este binario.



Tras la cabecera, el archivo sólo contiene todas esas palabras de CPU en secuencia.



4.1 Validaciones para archivos binarios de programa

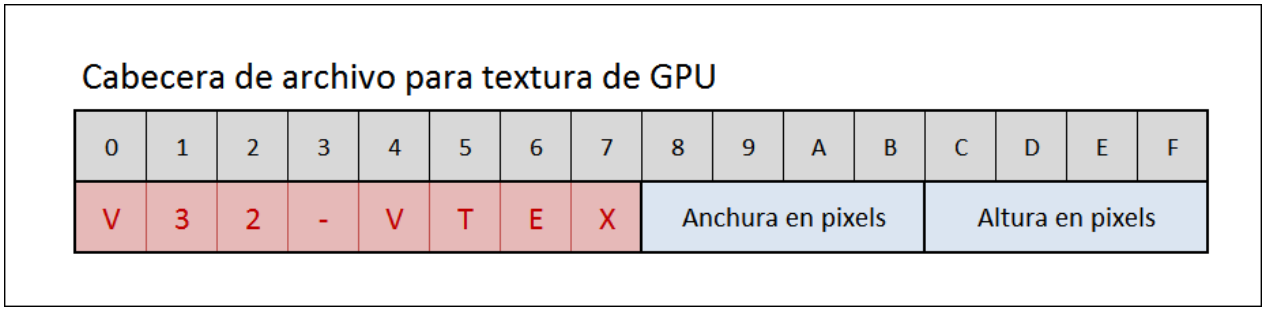
Además de verificar la firma de archivo correcta, se pueden hacer las siguientes comprobaciones para garantizar que un archivo binario de programa es válido:

1) El número de palabras declarado es válido. El rango va desde 1 hasta el límite de tamaño aplicable para la ROM de programa (ver límites para cartucho y BIOS en la sección 7).

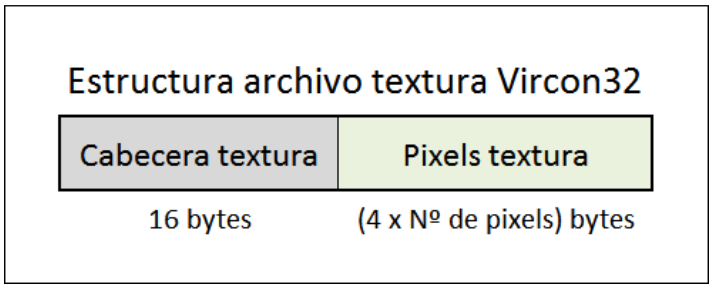
2) El tamaño del archivo en bytes debe ser 12 + 4 x { Nº palabras de datos }.

5 Archivos de textura GPU

Aunque el tamaño interno de una textura de GPU es fijo (1024 x 1024 pixels), la mayoría de las texturas de un juego sólo usarán parte de esa área. Por ello, los archivos de textura sólo almacenan un rectángulo de la anchura y la altura especificadas. Así, aparte de la firma del archivo, la cabecera de un archivo de textura GPU sólo contiene 2 campos enteros que indican las dimensiones en pixels del área de textura almacenada.



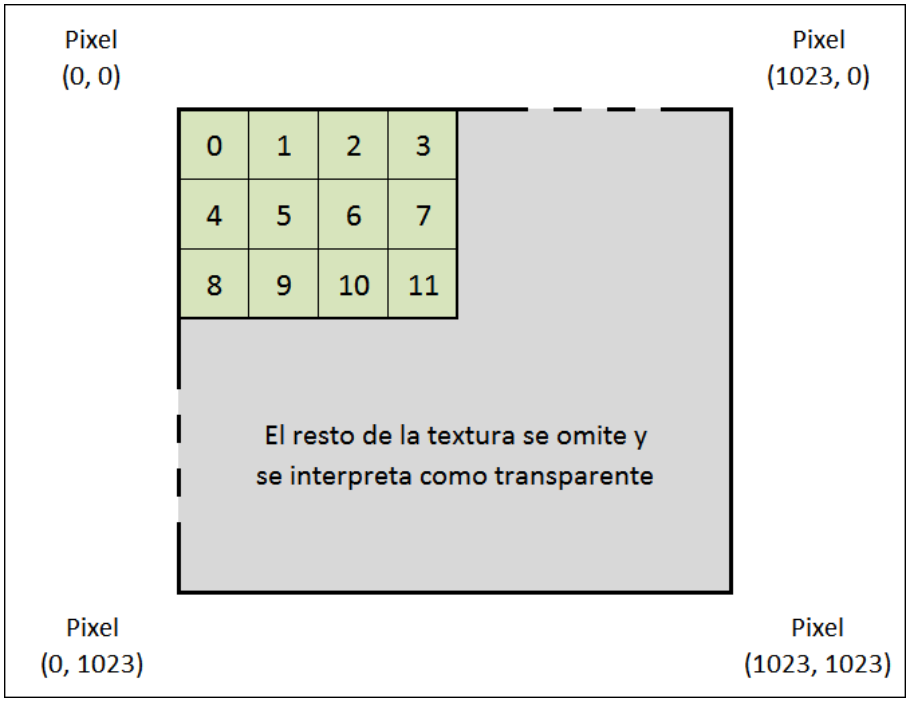
Tras la cabecera, el archivo sólo contiene todos esos pixels en secuencia.



5.1 Orden de almacenamiento de los pixels de la textura

El rectángulo de área de textura guardado se considerará la parte superior izquierda de la textura completa. Para el resto de los píxeles de la textura, hasta el tamaño completo, se considera que sus 4 componentes RGBA son cero (es decir, totalmente transparentes).

Los pixels se almacenan empezando por la esquina superior izquierda, y avanzando primero de izquierda a derecha y luego de arriba a abajo. Por ejemplo, para una textura que almacena 4 x 3 pixels, el orden de de esos 12 pixels será el que se muestra aquí.



5.2 Validaciones para archivos de textura GPU

Además de verificar la firma de archivo correcta, se pueden hacer las siguientes comprobaciones para garantizar que un archivo de textura GPU es válido:

1) El tamaño de textura declarado es válido. Para anchura y altura, el rango va de 1 a 1024. La anchura y la altura no tienen por qué ser iguales.

2) El tamaño del archivo en bytes debe ser 16 + 4 x { Anchura } x { Altura }.

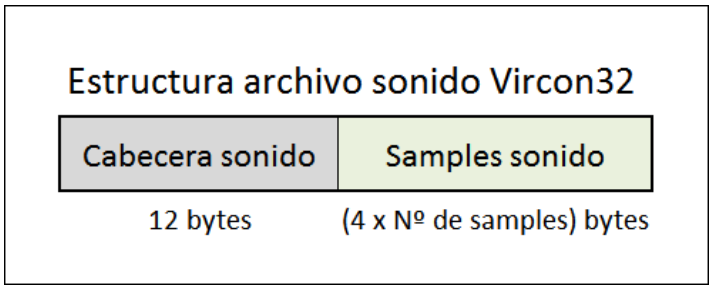
6 Archivos de sonido SPU

Aparte de la firma del archivo, la cabecera de un archivo de sonido SPU sólo contiene 1 campo entero que indica el número de samples de 32 bits que componen este sonido.

Cabecera de archivo para sonido de SPU

0	1	2	3	4	5	6	7	8	9	A	B
V	3	2	-	V	S	N	D	Nº samples de sonido			

Tras la cabecera, el archivo sólo contiene todos esas samples en secuencia. Vircon32 asume que todos los sonidos SPU tienen velocidad de reproducción de 44100 samples/s.



6.1 Validaciones para archivos de sonido SPU

Además de verificar la firma de archivo correcta, se pueden hacer las siguientes comprobaciones para garantizar que un archivo de sonido SPU es válido:

1) El número de samples declarado es válido. El rango va desde 1 al límite de tamaño de ROM de audio aplicable (ver límites para cartucho y BIOS en la sección 7).

2) El tamaño del archivo en bytes debe ser 12 + 4 x { Nº de samples }.

7 Archivos ROM de Vircon32

La estructura de un archivo ROM es más compleja que la de otros formatos. Esto también se refleja en su cabecera. Su tamaño es de 128 bytes y contiene los siguientes campos:

Cabecera de archivo para ROM de Vircon32

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00h	V	3	2	-	C	A	R	T	Versión de Vircon				Revisión de Vircon			
10h	T	H	E		B	I	G		E	X	C	I	T	I	N	G
20h		S	H	O	O	T	E	R		G	A	M	E		B	Y
30h		J	E	A	N		M	C	F	E	R	G	U	S	O	N
40h	!		(	P	R	O	T	O	T	Y	P	E		5	)	(0)
50h	Versión de la ROM				Revisión de la ROM				Número de texturas				Número de sonidos			
60h	Inicio ROM programa				Tamaño ROM Programa				Inicio ROM Video				Tamaño ROM Video			
70h	Inicio ROM Audio				Tamaño ROM Audio				*(8 bytes reservados para posible uso futuro)*							

X	Firma del archivo. 8 caracteres, sin terminación nula
X	Título de la ROM. 64 caracteres (hasta 63 + terminación nula)

Los cartuchos y las BIOS comparten este mismo formato de archivo. Se distinguen por sus firmas de archivo: los cartuchos usan **`V32-CART`**, y las BIOS usan **`V32-BIOS`**.

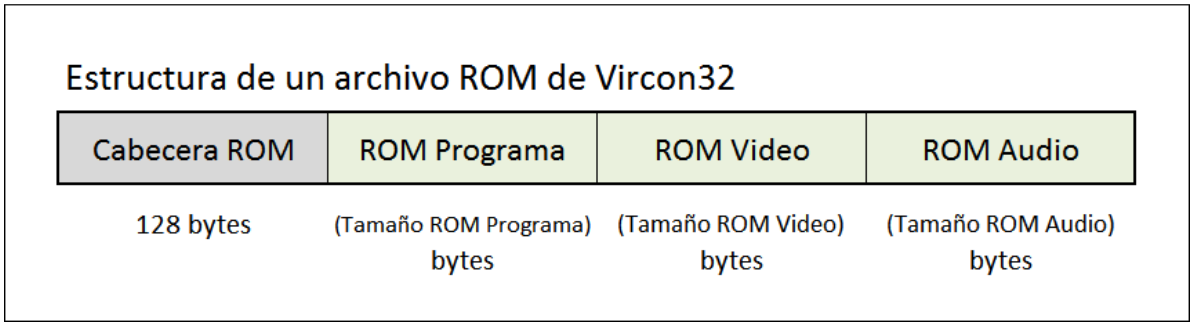
Los campos en azul (versiones y revisiones) son pares de números enteros, interpretados conjuntamente como un número de versión de la forma <versión . revisión>. La versión actual del estándar Vircon32 es 1.0 (la primera versión, y por tanto la única soportada).

El título y la versión de la ROM sólo se incluyen como metadatos de la ROM. Las implementaciones no están obligadas a procesar estos campos, ni siquiera a leerlos.

Los campos naranja (inicios y tamaños) también son pares de números enteros. Se usan para delimitar las 3 regiones dentro del archivo que corresponden a las 3 ROMs

empaquetadas juntas en un archivo ROM. Estos son los 3 componentes de un ejecutable: ROM de programa, ROM de video y ROM de audio.

Un archivo ROM puede considerarse como la unión de esos 3 componentes a la cabecera:



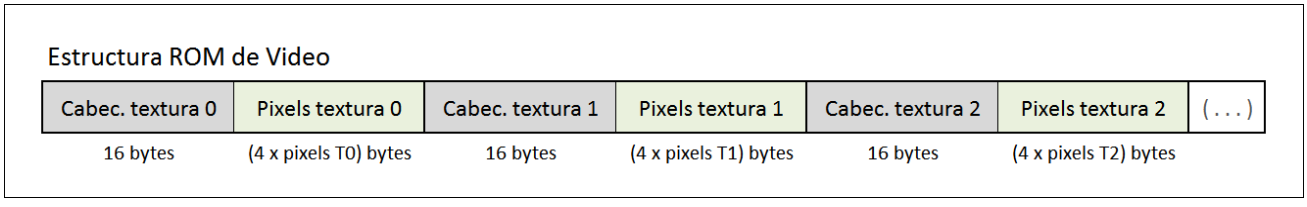
Los inicios y tamaños se expresan en bytes y, por tanto, siempre deben ser múltiplos de 4. Los inicios se expresan en número de bytes desde el principio del archivo. Por ejemplo: el inicio de la ROM de programa debe ser siempre 128, ya que está justo tras la cabecera.

Téngase en cuenta que, de estos 3 componentes, el único obligatorio es la ROM de Programa. Las ROMs de Video y/o Audio estarán ausentes si el número declarado de texturas y sonidos es cero, respectivamente.

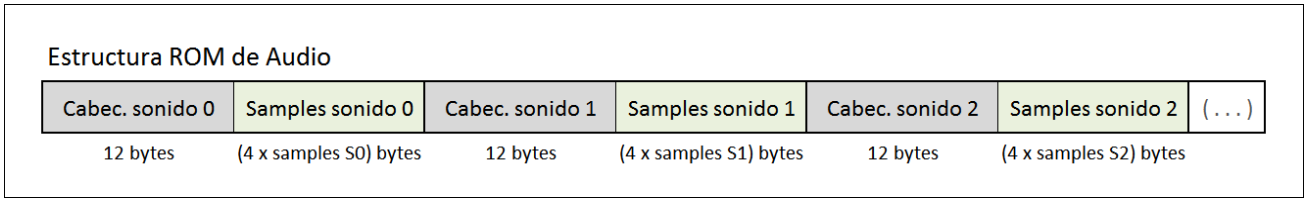
7.1 Estructura de las 3 componentes de las ROMs

Las 3 partes de un archivo ROM reutilizan los 3 formatos de archivo para activos descritos en secciones anteriores. Aquí, cada activo se guarda incrustando su archivo de activo individual en el archivo ROM. La estructura de la ROM de Programa será la misma que la de un archivo binario de programa (ya que siempre hay exactamente 1 binario).

Para la ROM de Video, su estructura será una secuencia de archivos de textura GPU almacenados uno tras otro:



Y de forma similar, la estructura de la ROM de Audio consistirá en una secuencia de archivos de sonido SPU colocados uno tras otro:



7.2 Validaciones para archivos ROM de cartucho

Además de verificar la firma de archivo correcta, se pueden hacer las siguientes comprobaciones para garantizar que un archivo ROM de cartucho es válido:

1) La versión y revisión de Vircon declaradas deben corresponder a versión 1.0.

2) El tamaño del archivo en bytes debe ser igual a: 128 + { Tamaño ROM Programa } + { Tamaño ROM Video } + { Tamaño ROM Audio }

3) El número de texturas declarado debe estar en el rango entre 0 y 256.

4) El número de sonidos declarado debe estar en el rango entre 0 y 1024.

5) Todas las texturas deben tener anchura y altura válidas, de 1 a 1024 pixels.

6) El binario existente debe tener un número válido de palabras, desde 1 hasta el límite de la ROM de programa para cartucho de (128 x 1024 x 1024) palabras.

7) Todos los sonidos deben tener un número válido de samples, desde 1 hasta el límite de la ROM de audio para cartucho de (256 x 1024 x 1024) samples.

8) La suma de los samples de todos los sonidos existentes no debe superar el límite de la ROM de audio para cartucho, de (256 x 1024 x 1024) samples.

7.3 Validaciones para archivos ROM de BIOS

Además de verificar la firma de archivo correcta, se pueden hacer las siguientes comprobaciones para garantizar que un archivo ROM de BIOS es válido:

1) La versión y revisión de Vircon declaradas deben corresponder a versión 1.0.

2) El tamaño del archivo en bytes debe ser igual a: 128 + { Tamaño ROM Programa } + { Tamaño ROM Video } + { Tamaño ROM Audio }

3) Tanto el número de texturas como de sonidos declarados deben ser 1.

4) La textura existente deben tener anchura y altura válidas, de 1 a 1024 pixels.

5) El binario existente debe tener un número válido de palabras, desde 1 hasta el límite de la ROM de programa para BIOS de (1024 x 1024) palabras.

6) El sonido existente debe tener un número válido de samples, desde 1 hasta el límite de la ROM de audio para BIOS de (1024 x 1024) samples.

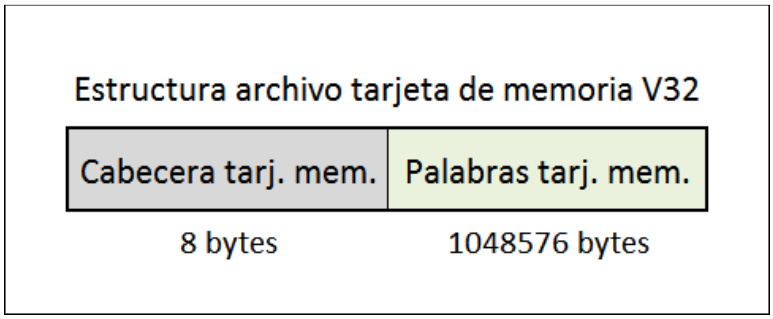
8 Archivos de tarjeta de memoria Vircon32

Para un sistema Vircon32 el contenido de una tarjeta de memoria es sólo un rango usable de posiciones de memoria, y su tamaño es siempre exactamente 256 KWords = 1MB. Por eso la cabecera de un archivo de tarjeta de memoria no necesita más información que la propia firma del archivo. Así, la cabecera de estos archivos es simplemente la siguiente:

Cabecera archivo tarjeta de memoria V32

0	1	2	3	4	5	6	7
V	3	2	-	M	E	M	C

Tras la cabecera, el archivo sólo contiene todas las palabras CPU de 32 bits de la tarjeta en secuencia.



Como recordatorio, las 20 primeras palabras del contenido de la tarjeta se consideran la "firma del juego". Aunque estas palabras se tratan igual que el resto de la tarjeta, los juegos suelen usarlas para identificar el contenido de cada tarjeta. Así se evita que un juego cargue datos destinados a otro (con formatos de guardado incompatibles).

8.1 Validaciones para archivos de tarjeta de memoria

Además de verificar la firma de archivo correcta, se pueden hacer las siguientes comprobaciones para garantizar que un archivo de tarjeta de memoria es válido:

1) El tamaño del archivo en bytes debe ser igual a 1048584.

(Fin de la parte 9)