19

CONSTRUCCION DE UN MULTIPROCESADOR QUE MANEJA LISP
PARTE II: ARQUITECTURA DE LA MAQUINA AHR

Adolfo Guzmán
Luis Lyons
Luis Hugo Peñarrieta
David Rosenblueth
Raul Gómez
Manuel Correa
Dora Gómez
Norma A. de Rosenblueth
Juan López
Isauro Morales
Pablo Martínez
Departmento de Sistemas de Cómputo, IIMAS
Universidad Nacional Autónoma de México

Sumario

Este artículo es continuación y complemento del artículo "Construcción de un Multiprocesador que Maneja Lisp, Parte I: Teoría de Operación de la Máquina AHR", que aparece en estas mismas memorias.

El obje tivo principal de este proyecto es construir una máquina de procesamiento en paralelo para lenguajes aplicativos.

Otro obje tivo es proveer un sistema para enseñanza e investigación en ciencias de la Computación, específicamente para lenguajes en paralelo, sistemas distribuidos, sistemas heterárquicos y arquitectura de máquinas.

Sucede a veces que la enseñanza de las Ciencias de la Computación, sobre todo en las disciplinas arriba mencionadas, se vuelve "de papel" solamente, en el sentido de que los estudiantes carecen de un sistema real (máquinas y programas) donde ensayar y practicar, donde desarrollar programas en paralelo, y en general donde se puedan efectuar mediciones y comprobaciones de cambios en la arquitectura, en sus parámetros o en la programación.

En cuanto a la investigación, el Proyecto AHR pretende ser un lugar donde se puedan realizar diseños en los que se amalgamen fuertemente sistemas digitales (hardware), microprogramados tal vez, con sistemas de operación (software). Es decir, cubre el área donde es válido adaptar o cambiar la máquina para mayor provecho de los programas, además de adaptar éstos a la máquina.

Arquitectura de la Máquina AHR

Visión global de los circuitos

La máquina AHR puede descomponerse en las partes mostradas en la figura "Las partes de la Máquina AHR". La operación de la máquina es como sigue:

El usuario mete su programa por el teletipo, y es almacenado en el disco. Esta interacción la desarrolla el procesador de entrada y salida. Cuando un programa de Lisp va a ser ejecutado, es depositado por el procesador de e/s, a través de su ventana, en la memoria pasiva. El distribuidor inicia la ejecución, según se describió anteriormente en la Parte I.

Las memorias pasivas y de variables se accesan a través de sus controladores respectivos, los que trabajan en conjunción con sus árbitros. El FIFO del distribuidor contiene apuntadores a nodos listos para ser evaluados por los procesadores de Lisp, en tanto que el fifo-hardware o $FIFO^2$ contiene los objectos-dato listos a ser impresos (por el procesador de e/s).

Un procesador de Lisp convierte nodos listos a evaluarse en objectos-dato de Lisp. Estos son colocados en la parrilla para su reintegración al proceso de evaluación en paralelo.

Un procesador de Lisp contiene un microprocesador (Z80) con su memoria privada (eprom y ram), además de una memoria compartida ram* que es accesada tanto por el microprocesador como por el distribuidor y los controladores.

La computadora AHR puede contener hasta 64 procesadores de Lisp.

Los procesadores de Lisp pueden comunicarse con el procesador de entrada/salida a través de un canal lento.

Procesadores de Lisp (cajas)

Son los elementos activos del sistema, puesto que en ellos se ejecutan las funciones primitivas del lenguaje de alto nivel; en este caso Lisp. Asimismo pueden ser "especializadas" de manera que se utiliza un hardware especialmente orientado a la ejecución de determinadas funciones. Típicas: multiplicador de punto flotante, extractor de raíces cuadradas, correlacionador.

Cada procesador tiene su RAME (ram de especialización), que es una memoria de 256 posiciones de un bit. Cada posición corresponde a una función de Lisp. Las cajas deben inicializar RAME escribiendo un 0 o un 1 en cada una de las posiciones, 1 en las posiciones correspondientes a las funciones que puede realizar y 0 en las que no. Después de haber sido iniciado, el contenido de RAME no cambia, a menos que se haya reconfigurado al procesador de Lisp correspondiente.

Nota: como ya se dijo antes, las Versiones 0 y 1 de la máquina AHR no son reconfigurables, por lo que carecen de

Artículo 50

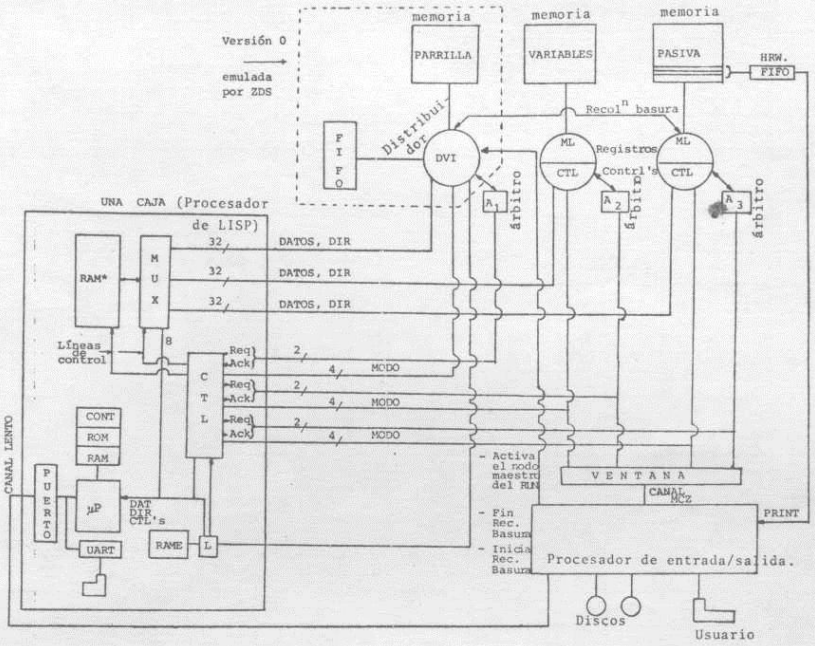


FIGURA "L A S P A R T E S D E L A M A Q U I N A A H R"

ESPECIFICACIONES	V E R S I O N C E R O		VERSION UNO
	V0-	V0+	V1
Fecha de terminación	Oct. 80	Feb. 81	Dic. 81
CPU	Z80	Z80 o Z80 A	Z80 A o Z80
Memoria ram	16 Kby expand. a 48 kby.	16 kby	4 kby
Memoria EPROM	1 kby expandi. a 4 kby.	2 a 8 kby.	1 ó 2 kby
Reloj	2.5 a 4 Mhz	2.5 ó 4 Mhz	4 Mhz
Bus (canal) lento	16 bits RS-232 C	16 bits RS-232 C	16 bits RS-232 C
MEMORIA PASIVA	6 kpal de 32 b 2 kpal de 22 b t_{acc}= 55 nanoseg (expandible a 2 a la 19 pal)	20 kpal de 22 b (listas) 4 kpal de 20 b (átomos) t_{acc}= 55 ns	32 kpal de 22 b (listas) 16 kpal de 20 b (átomos) t_{acc}= 150 ns. chequeo paridad.
MEMORIA DE VARIABLES	4 kpal de 32 b (expandible a 2 a la 19 pal = 2 Megabytes) t_{acc}= 55 ns.	16 kpal de 32 b	16 kpal de 32 b t_{acc}= 150 ns chequeo paridad.
PARRILLA	(ZDS 1/25; qde) 1 k nodos de 32 by (8 kpal de 32 b) t_{acc}= lento.	ZDS 1 knodos de 32 by (8 kpal de 32 bits)	8 kpal de 32 bits t_{acc}= 55 ns (expandible hasta 2 a la 19).
FIFO	ZDS 1/25. 128 apunt. de 16 bits c/u	ZDS 128 apunt de 16 b	4 kpal de 19 bits tacc= 55 ns (exp. a 2 a la 19 pal)
DISTRIBUIDOR	lento (emulado en la ZDS 1/25)	lento	rápido (por circuitos)

TABLA "E S P E C I F I C A C I O N E S"

La versión 0 tiene dos etapas, V0- y V0+, y estará lista y funcionando en febrero de 1981. La versión 1 desarrolla la velocidad final de la arquitectura heterárquica propuesta.

RAME. Es decir, los procesadores de Lisp de estas versiones no pueden ser especializadas. Veáse la Comunicación AHR-76-1.

Los microprocesadores Z80 forman parte de los procesadores de Lisp, que además tienen memoria dinámica de 16k bytes (expandible a 48k), memoria EPROM de 4k bytes, dos puertos paralelos de 8 bits y un canal asíncrono/síncrono.

Parrilla (memoria activa)

El programa a ejecutarse se encuentra en la parrilla, parcial o totalmente, estructurado como un árbol de nodos con el número de descendientes inmediatos limitado a cuatro. El desarrollo de algunas ramas está condicionado por el proceso mismo. Cada nodo representa una función de Lisp, y cada hijo corresponde a argumentos que han de ser evaluados antes de poder ejecutar la operación indicada por el padre. De manera que sólo están en condiciones de ser evaluados aquellos nodos que en un momento dado son nodos terminales (número de hijos no evaluados = 0).

Teóricamente se pueden evaluar en paralelo tantos nodos terminales como existan en un momento dado, limitado en la práctica únicamente por el número de procesadores de Lisp disponibles.

Memoria Pasiva

Contiene programas, datos y resultados que no están en la parrilla. Los nódos en la parrilla hacen referencia a esta memoria a través del campo CODLIG (liga al código). Durante el proceso de transformación del programa a notación en forma de nodos en la parrilla, se hace referencia constante a esta memoria. La memoria pasiva tiene un espacio de listas, con circuitería especial (ver controlador de pasiva) para manejo eficiente de este tipo de estructuras, y un espacio de memoria contigua para manejo de arreglos. Aquí se encuentra una dirección ficticia para accesar al fifo^2 (véase éste) que es el puerto de entrada y salida.

La memoria pasiva puede accesarse en modo palabra, en modo celda (Multibyte) o en modo ventana (byte a byte). La construcción de la memoria pasiva es bastante densa: 128k bytes por tarjeta. (Ver Figura "Memoria Pasiva")

Espacio de listas: Cada nodo de una lista ocupa 2 palabras: la primera es el CAR, la segunda el CDR.

CAR - Los bits 0 y 1 indican el tipo de información que hay en los siguientes 20 bits (es decir, AHR es una arquitectura con datos etiquetados(17)):

Bits 0-1:	Tipo de información en los sig. 20 b:
0 0	Un apuntador a una lista.
0 1	Atomo literal (normal).
1 0	Atomo entero.
1 1	Atomo número real (apuntador a

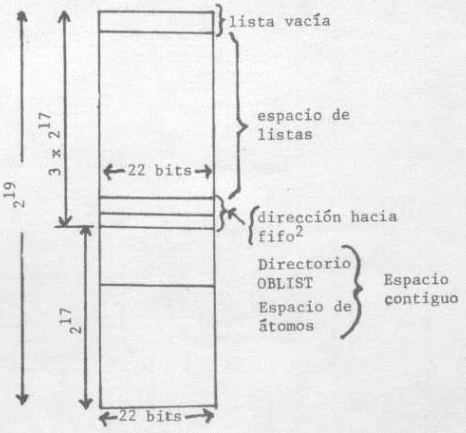


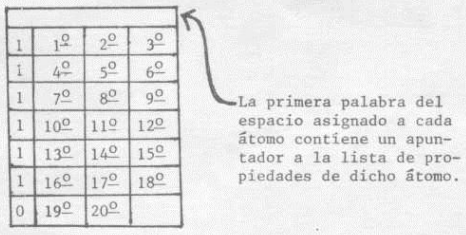
FIGURA "MEMORIA PASIVA"

la parte baja de la memoria de variables),

CDR - Los bits 0, 1, y 2 son para recolección de basura. Los 19 bits restantes son apuntadores a listas. Esta dirección siempre es par pues apunta al CAR de otro nodo.

Espacio contiguo: Contiene espacio para el directorio (oblist) y para átomos. Existe un oblist para cada usuario y aquí se encuentra la cabeza del oblist de cada uno, apuntando al espacio de listas.

La longitud del identificador de un átomo (print name) no está limitada; los caracteres están codificados en ASCII. Caben tres caracteres por palabra. El bit 0 cuando es 0 nos indica terminación de cadena; cuando es 1 indica que la cadena continúa.



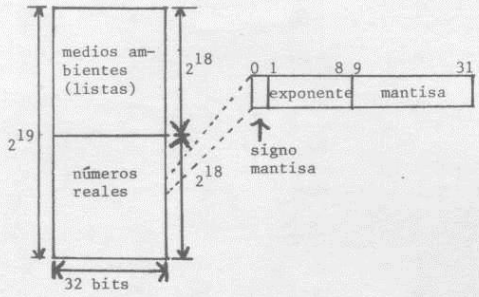
Memoria de variables

La memoria de variables contiene los valores de los diferentes átomos (variables) que se están usando en un determinado momento en la

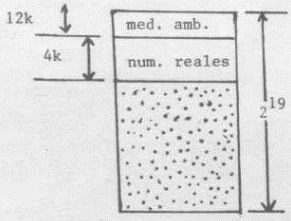
ejecución de un programa.

Por ser la ejecución en paralelo, una misma variable puede tener varios valores simultáneamente; es perfectamente factible que A valga 5, 4 y (M N Q) a la vez.

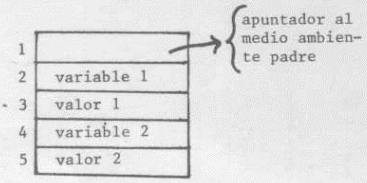
La organización de la memoria de variables se muestra en la siguiente figura:



En la versión VO+ sólo hay 4k de memoria para los números reales, así:



Los medios ambientes son listas de "celdas" o grupos de 5 palabras de 32 bits, cada celda es así:

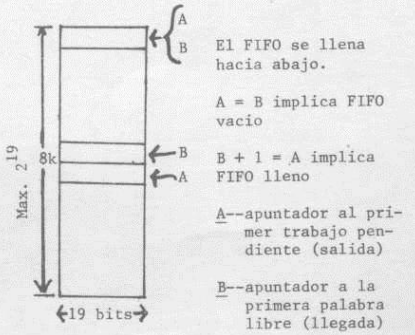


La memoria de variables puede accesarse en modo palabra, en modo nodo (multibyte), o en modo ventana (byte a byte). La construcción de la memoria de variables es bastante densa: 128 kilobytes por tarjeta.

FIFO (pizarrón)

Apunta a nodos en la parrilla que están listos para ser evaluados. La condición de nodo listo se determina cuando el campo NANE (núm. de argumentos no evaluados) es igual a cero. Todo nodo una vez evaluado, le resta un 1 al NANE de su padre y cuando éste toma el valor cero, se anota al nodo padre en el pizarrón (FIFO). El distribuidor (véase éste) toma trabajos del FIFO para entregárselos a los procesadores de Lisp que están haciendo peticiones de trabajo (petición de burocracia de entrada).

El pizarrón o FIFO es una lista circular del tipo first in first out (FIFO), de manera que los trabajos ahí anotados se atienden por orden de llegada. El FIFO cuenta con dos registros A y B. A apunta al primer trabajo pendiente; B, a la primera celda libre.



Acoplador

Un acoplador es la interfaz entre los procesadores de Lisp y los canales de la máquina AHR. El acoplamiento se hace bajo el principio de "buzon". Posee una memoria de 16 palabras de 32 bits, con tiempo de acceso de 60 nanosegundos.

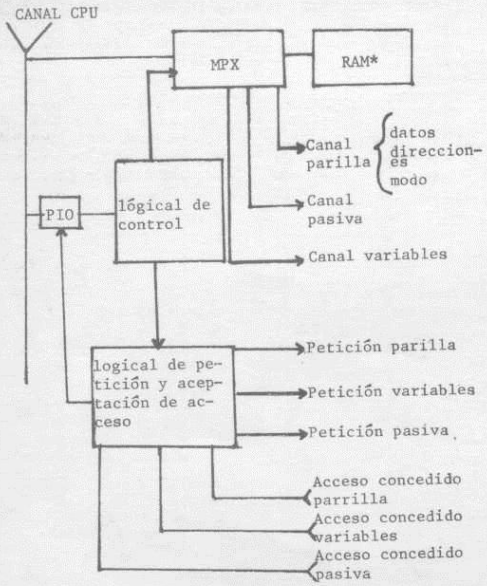
Arbitros

Hay tres árbitros, los que actúan independientemente: uno para la memoria pasiva, otro para la memoria de variables, y otro para la parrilla.

Un árbitro es un circuito que concede el acceso a un recurso (memoria en este caso) cuando varios usuarios (procesadores de Lisp) requieren uso compartido de él. Si un procesador de Lisp pide acceso a la memoria de variables, por ejemplo, el árbitro de la memoria de variables le concede acceso a ella. Pero si dos procesadores de Lisp piden acceso simultáneo a la memoria de variables, el árbitro de esta memoria concede acceso a uno de ellos primero y luego al otro.

Esta función de arbitraje es necesaria porque dos procesadores no pueden usar el mismo canal de acceso a la memoria al mismo tiempo: se formaría una confusión, habiendo interferencia en la información.

Funcionamiento del árbitro: cada procesador de Lisp tiene dos conexiones al árbitro, una para pedirle acceso al recurso (a la memo-



A C O P L A D O R

Este circuito conecta el procesador de Lisp con el resto de la máquina AHR.

ria respectiva), y otra por donde el árbitro le responde, dándoselo o negándoselo.

Cada procesador de Lisp tiene una prioridad preestablecida, según su posición en el canal rápido (digamos que están numerados 1, 2, 3, . . .64 con respecto a su prioridad), de tal suerte que el árbitro escoge siempre al procesador de más alta prioridad, entre todos los que estén solicitando acceso cuando el controlador o distribuidor se desocupen.

Cada árbitro es asíncrono; los procesadores de Lisp pueden solicitar acceso en cualquier momento a cualquier memoria.

Cada árbitro tarda 400ns en responder, y arbitra hasta 64 procesadores.

Procesador de entrada y salida ("caja inteligente")

En este procesador (actualmente un MCZ que contiene un Z80) corre un sistema operativo normal que maneja a los teletipos del usuario, discos e impresoras.

El procesador de entrada y salida accesa las memorias de variables y la pasiva por medio de una ventana que adelante se describe. El procesador de e/s no tiene acceso directo a la parrilla.

Las tareas principales del procesador de e/s son:

* atender al usuario, leerle sus comandos y datos e imprimirle sus resultados.
* Manejar los archivos del usuario, en disco.
* Inicia el Sistema AHR.
* Cargar a la memoria pasiva, a través de la ventana, los programas de Lisp que vayan a ejecutarse.
* Activar el nodo maestro para iniciar la ejecución de un programa.
* Le da muerte a un programa.
* Inicializa la recolección de basura.
* Finalizar la recolección de basura.
* Actualmente, el recolector de basura se ejecuta en el procesador de e/s.

Veán también las secciones "Recolector de Basura", "Programas del Procesador de e/s", "Inicialización del sistema", "Comunicación a través del canal lento", y "Manejo e impresión de errores", en el Capítulo "Sistema de Programación".

Ventana

La ventana permite al procesador de e/s accesar las memorias de variables y la pasiva. (Ver AHR-80-13).

De los 64k de direccionamiento del procesador de e/s, la ventana está colocada en los terceros 16k (ver fig. 'Ventana'). Esto es, cualquier dirección de la 1000 0000 0000 0000 = 8000 a la 1011 1111 1111 1111 = BFFF inclusive se mapea a la ventana.

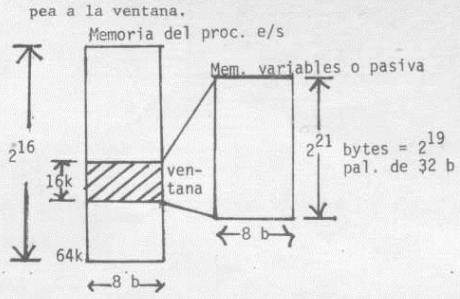
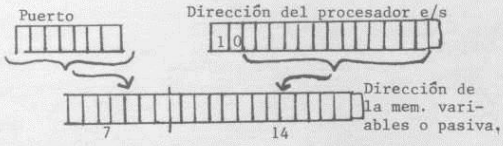


FIGURA "V E N T A N A"

A través de los 16 bits de direccionamiento de la memoria, de los que se usan catorce más siete que salen por un puerto del procesador, se forma una dirección de 21 bits para accesar cualquier byte de la memoria de variables o de la pasiva.

Los dos primeros bits son 1 y 0 (que indican el tercer cuarto de memoria) y están fijos. Los siguientes 14 bits de la dirección del procesador de e/s pasan a formar los 14 bits menos significativos de la dirección de la memoria de variables (o de la pasiva), en tanto que por un puerto el procesador de e/s saca otros siete bits, que pasan a ser los 7 bits más significativos de una dirección de 21 bits, lo que permite accesar 2^{21} bytes o 2^{19} palabras de 32 bits. El direccionamiento es por bytes en el modo ventana.

Para accesar un byte de la memoria pasiva (o de la de variables) desde el procesador e/s se utilizan, pues, varias instrucciones: escribir en el puerto, formar la dirección y solicitar acceso al canal correspondiente.



Mientras el procesador de e/s está accesando alguna memoria en modo ventana, los procesadores de Lisp encuentran cerrado el acceso a esa memoria. Es hasta que el procesador de e/s finaliza su acceso en modo ventana (y se lo comunica al controlador con la señal "fin de uso de ventana") cuando los procesadores de Lisp reanudan su acceso a la memoria.

La ventana se hizo para transmitir datos a velocidades de CPU entre el procesador de e/s y las memorias.

Distribuidor

Permite la comunicación entre los procesadores de Lisp, el FIFO y la parrilla. La velocidad a la que opera es crítica para la eficiencia de la máquina. Para la versión 0 las funciones del distribuidor están desarrolladas en programas y para la versión 1 estas funciones serán ejecutadas en circuitos.

Las funciones del distribuidor son:

= crear un nodo maestro en la parrilla y anotarlo en el FIFO cuando se desea ejecutar un programa.

= suspender la ejecución de los procesadores de Lisp, cuando hay recolección de basura.

= atender las peticiones de los procesadores de Lisp en los modos que se describen a continuación.

Controladores

Hay dos controladores, uno para la memoria pasiva y otro para la de variables. Ambos se componen de registros y controles. Por medio de los registros se implementan las transferencias de datos requeridos para realizar las funciones de los controladores, quienes ordenan estas transferencias.

Los controladores permiten el acceso inteligente a las memorias, en varios modos.

Controlador de la Memoria de Variables

Características funcionales del controlador:

- acceso aleatorio.
- cabeza de stack.
- dirección de la. palabra disponible para números (alfa) para recolección de basura.
- mantenedor de listas.

Modos

Modo 1. (Leevar). Lectura aleatoria de una sola palabra de 32 bits.
- incluye acceso a alfa
- incrementa alfa
- existe una localidad "beta" (por pianitos). Si alfa = beta se prende la línea de prerecolección de basura. Normalmente, beta = número total de localidades - 100.

Modo 2. (Escvar). Escritura aleatoria de una sola palabra.

Modo 3. (Ligvar). Ligar nodos (5 palabras contiguas de 32 bits). Se regresan nodos, que se ligan a la lista de nodos libres.

Modo 4. (Celvar). Petición de nodos (desensarta nodos). El mantenedor de listas entrega nodo libre (5 palabras de 32 bits c/u, contiguas).

Modo 5. (Lemvar). Lectura múltiple (5 palabras de 32 bits): lee un nodo.

Modo 6. (Esnvar). Escritura múltiple (5 palabras de 32 bits): escribe un nodo.

Modo 7. (Venvar). Modo ventana.

Nótese que en la memoria de variables, un nodo o celda son 5 palabras contiguas de 32 bits c/u.

Controlador de la memoria pasiva

Los modos de este controlador son los siguientes:

Modo 1. (Leepas). Lectura aleatoria de una

palabra de 22 bits.

Modo 2. (Escpas). Escritura aleatoria de una palabra de 22 bits.

Modo 3. (Ligpas). Ligar listas (liberar 2 palabras de 22 bits, o sea un nodo).

Modo 4. (Celpas). Petición de nodo libre (2 palabras de 22 bits).

Modo 5. (Lempas). Lectura múltiple a pasiva (2 palabras de 22 bits).

Modo 6. (Esmpas). Escritura múltiple a pasiva (2 palabras de 22 bits).

Modo 7. (Venpas). Modo ventana.

Nótese que en la memoria pasiva, un nodo o celda son dos palabras contiguas de 22 bits c/u.

Conclusiones y Recomendaciones

El esfuerzo del Proyecto AHR ha hecho posible que el IIMAS-UNAM y la comunidad académica cuenten con una computadora de tamaño mediano, de propósitos generales, diseñada y construida en México y por técnicos nacionales.

Se ha logrado un entendimiento teórico y práctico de importantes efectos en Computación; se conoce cómo administrar, coordinar, controlar y supervisar la ejecución de varias corrientes de instrucciones (instruction streams), con un control hecho en su mayoría por circuitos.

La manufactura de la Computadora AHR muestra que es posible procesar programas de propósito general (que no han sido escritos explícitamente para un ambiente de paralelismo) en un multiprocesador con bastante simultaneidad.

Se ha aprendido cómo construir computadoras orientadas a lenguaje de máquina de alto nivel.

Avances en Paralelismo

* Se tiene una herramienta poderosa para la experimentación con lenguajes y sistemas de programación en paralelo.

* Se creó, desarrolló y encontró útil el concepto de heterarquía.

* Se sabe cómo administrar, coordinar, controlar y usar los recursos necesarios para llevar a cabo un cómputo por varias computadoras simultáneamente.

* Se conoce cómo ejecutar un programa de propósito general en paralelo, sin instrucciones ni comandos específicos de parte del programador.

* Se ha formado un grupo de personas capaces de proseguir investigaciones de frontera en procesamiento distribuido, procesamiento en paralelo, bases de datos por hardware, etc.

Bibliografía

Informes Tecnicos del Laboratorio AHR

Serie y No.	AUTOR	TITULO	AÑO
NA 133	Adolfo Guzmán Raymundo Segovia	AHR-76-1 A parallel configurable Lisp machine.	1976
NA 200	David Rosenblueth Carlos Verlarde	AHR-79-2 La máquina AHR para procesamiento en paralelo, la. etapa.	1979
NA 214	Kermer Norkin Dora Gómez	A new description for data transformation in the AHR computer	
NA 215	Kermer Norkin David Rosenblueth	AHR-79-5 Towards optimization in AHR.	1979
NA 216	Adolfo Guzmán	AHR-79-6 La computación distribuida como una alternativa del futuro.	1979
AM 15	Adolfo Guzmán	AHR-80-7 Ciento veinticinco proyectos y temas de tesis en computación.	1980
NA 229	Luis Hugo Peñarrieta	AHR-80-9 Detección de errores en la máquina AHR.	1980
NA 253	Adolfo Guzmán, Luis Lyons, et al.	AHR-80-10 La computadora AHR: Construcción de un multiprocesador con Lisp como lenguaje principal.	1980
VE 17	Victor Germán Sánchez	AHR-80-12 Sistema de Información General de estructura reconfigurable	
NA 246	Adolfo Guzmán	AHR-80-13 A parallel heterarchical machine for high level language processing.	1980