

CHƯƠNG II

LẬP TRÌNH NHÚNG CĂN BẢN

Lời đầu chương

Lập trình nhúng là một vấn đề tương đối khó, đòi hỏi người nghiên cứu học tập phải có sự kiên nhẫn và lòng đam mê thật sự mới có thể hiểu và tiến xa trong lĩnh vực này. Với mục đích định hướng cho người học có sự đam mê ban đầu trong môn lập trình hệ thống nhúng, nhóm thực hiện đề tài đã trình bày một cách chi tiết những nội dung cốt yếu và cần thiết sao cho người học có thể ứng dụng ngay những gì đã học vào thực tế, tránh trường hợp lý thuyết suông quá tổng quát, nhằm tạo sự hứng thú trong quá trình học tập.

Trong chương II Lập trình nhúng căn bản, như tên gọi của nó, chương này trình bày những kiến thức rất căn bản về phần cứng hệ thống nhúng có liên quan đến kit thí nghiệm KM9260 như: Vi điều khiển AT91SAM9260, cấu trúc các linh kiện, hệ điều hành nhúng linux, các phần mềm hỗ trợ lập trình, ...

Để nghiên cứu những nội dung trong chương này đạt hiệu quả cao nhất, người học phải có những kiến thức và kỹ năng nền tảng sau:

- Kiến thức tổng quát về kỹ thuật số, lý thuyết vi xử lý, hiểu và sử dụng thành thạo một loại vi xử lý đơn giản (chẳng hạn 89x51, PIC, ...).
- Sử dụng thành thạo những thao tác cơ bản trong PC tương đương trình độ A.
- Kiến thức tổng quát về lý thuyết hệ thống nhúng, hệ điều hành, ...

Chương này được trình bày theo cấu trúc từng bài học, mỗi bài học có thể ngắn hoặc dài tùy theo nội dung kiến thức trình bày. Nội dung kiến thức trong mỗi bài là một vấn đề riêng biệt nhằm giúp cho người học lĩnh hội một cách chắc chắn vai trò của bài học trong quá trình tìm hiểu nghiên cứu lập trình chương trình ứng dụng trong hệ thống nhúng.

Cấu trúc các bài học bao gồm:

- **Bài 1:** Tổng quan về vi điều khiển AT91SAM9260;

Bài này cung cấp cho người học những kiến thức về vi điều khiển ARM, sơ đồ chân, bản đồ vùng nhớ của vi điều khiển sử dụng trong kit KM9260. Đây là những kiến thức quan trọng, giúp cho người học hiểu được nguyên lý kết nối phần cứng, cách sử dụng và khai báo địa chỉ vùng nhớ trong phần mềm, ... được đề cập trong những bài học khác.

- **Bài 2:** Một số thao tác cơ bản trên hệ điều hành Linux;

Hệ điều hành Linux được trình bày trong bài này là một hệ điều hành dành cho máy tính PC và cũng có một số đặc điểm giống với hệ điều hành chạy trên hệ thống nhúng. Mục đích là cung cấp cho người học những kiến thức về quản lý bộ nhớ, các thao tác từ đơn giản đến phức tạp trong môi trường shell của hệ điều hành Linux và một số các phần mềm hỗ trợ biên soạn chương trình ứng dụng khác.

- **Bài 3:** Hệ thống phần cứng và phần mềm nhúng trong kit KM9260;

Bài này giới thiệu một số vấn đề về cấu trúc phần cứng và phần mềm trong kit KM9260 nhằm giúp cho người học hiểu rõ tên, vị trí, vai trò của từng linh kiện, sơ đồ chân kết nối, ... bên cạnh đó là nguyên lý và vai trò hoạt động của từng thành phần trong hệ thống phần mềm nhúng, ... Những kiến thức này có vai trò quan trọng trong lập trình giao tiếp phần cứng và sửa chữa hệ thống trong quá trình hoạt động nếu phát sinh ra lỗi.

- **Bài 4:** Phần mềm hỗ trợ lập trình nhúng;

Cung cấp cho người học kỹ năng sử dụng những chương trình hỗ trợ quá trình lập trình nhúng như: Chương trình máy tính ảo, chương trình hỗ trợ nạp chương trình thông qua cổng USB, thiết bị hiển thị đầu cuối qua cổng COM, phần mềm giao tiếp mạng. Các phần mềm này hỗ trợ việc đăng nhập, lập trình, biên dịch, và thực thi chương trình ứng dụng trong kit.

- **Bài 5:** Thao tác trên phần mềm hệ thống nhúng;

Sau khi nghiên cứu Bài 4, người học sẽ thực hành sử dụng những thao tác đã được học để thao tác biên dịch và nạp các thành phần trong hệ thống phần mềm nhúng bao gồm rootfilesystem, uboot, kernel, driver và phần mềm ứng dụng để các thành phần này có thể chạy ổn định trên phần cứng hệ thống nhúng kit KM9260.

Những vấn đề được trình bày trong chương này không nhằm giải thích chuyên sâu về nguyên lý hoạt động của hệ thống nhúng mà cung cấp cho người học những kiến thức cụ thể có liên quan đến kit KM9260, phục vụ cho việc lập trình các ứng dụng thực tế giao tiếp giữa kit với các thiết bị ngoại vi trong chương sau.

Do tính riêng biệt trong từng bài học, nên người học có thể bỏ qua một số bài khi cảm thấy đã nắm vững các nội dung được giới thiệu trong mỗi bài học. Hoặc có thể làm tài liệu tham khảo khi chưa nắm vững một nội dung nào đó.

Sau đây chúng tôi sẽ trình bày lần lượt từng nội dung đã được liệt kê.

BÀI 1**TỔNG QUAN VỀ VI ĐIỀU KHIỂN
AT91SAM9260**

Thành phần quan trọng nhất cấu tạo nên một hệ thống nhúng chính là vi xử lý, vi điều khiển. Vi xử lý, vi điều khiển quyết định đến tính năng, hiệu quả ứng dụng của hệ thống nhúng. Vì vậy trước khi nghiên cứu về hệ thống nhúng thì chúng ta phải có một kiến thức căn bản về vi xử lý của hệ thống nhúng.

Kit nhúng KM9260 được xây dựng dựa trên nền vi điều khiển AT91SAM9260. Là vi điều khiển 32 bit thuộc họ ARM9 tiết kiệm năng lượng.

Trong phần này sẽ trình bày sơ lược các kiến thức về AT91SAM9260. Vi lập trình cho hệ thống nhúng không đặt nặng kiến thức về cấu trúc của vi xử lý. Nên trong phần này chỉ trình bày những kiến thức cần thiết cho quá trình sử dụng kit sau này như: sơ đồ chân của AT91SAM9260, nguồn cung cấp cho AT91SAM9260, bản đồ quản lý vùng nhớ của AT91SAM9260.

Trong vi điều khiển có tích hợp nhiều module ngoại vi khác nhau, kiến thức về từng module sẽ được trình bày trong mỗi bài thí nghiệm với các thiết bị ngoại vi khác nhau.

I. Họ vi điều khiển ARM:

Cấu trúc ARM (viết tắt từ tên gốc là **Acorn RISC Machine**) là một loại cấu trúc vi xử lý 32bit kiểu RISC được sử dụng rộng rãi trong các thiết kế nhúng. Do có đặc điểm tiết kiệm năng lượng, các bộ CPU ARM chiếm ưu thế trong các sản phẩm điện tử di động, mà với các sản phẩm này việc tiêu tán công suất thấp là một mục tiêu thiết kế quan trọng hàng đầu.

Ngày nay, hơn 75% CPU nhúng 32bit là thuộc họ ARM, điều này khiến ARM trở thành cấu trúc 32bit được sản xuất nhiều nhất trên thế giới. CPU ARM được tìm thấy khắp nơi trong các sản phẩm thương mại điện tử, từ thiết bị cầm tay (PDA, điện thoại di động, máy đa phương tiện, máy trò chơi cầm tay, và máy tính cầm tay) cho đến các thiết bị ngoại vi máy tính (ổ đĩa cứng, bộ định tuyến để bàn.) Một nhánh nổi tiếng của họ ARM là các vi xử lý Xscale của Intel.

Việc thiết kế ARM được bắt đầu từ năm 1983 trong một dự án phát triển của công ty máy tính Acorn.

Nhóm thiết kế, dẫn đầu bởi Roger Wilson và Steve Furber, bắt đầu phát triển một bộ vi xử lý có nhiều điểm tương đồng với Kỹ thuật MOS 6502 tiên tiến. Acorn đã từng sản xuất nhiều máy tính dựa trên 6502, vì vậy việc tạo ra một chip như vậy là một bước tiến đáng kể của công ty này.

Nhóm thiết kế hoàn thành việc phát triển mẫu gọi là ARM1 vào năm 1985, và vào năm sau, nhóm hoàn thành sản phẩm “thực” gọi là ARM2. ARM2 có tuyến dữ liệu 32bit, không gian địa chỉ 26bit tức cho phép quản lý đến 64 Mbyte địa chỉ và 16 thanh ghi 32bit. Một trong những thanh ghi này đóng vai trò là bộ đếm chương trình với 6 bit cao nhất và 2 bit thấp nhất lưu giữ các cờ trạng thái của bộ vi xử lý.

Có thể nói ARM2 là bộ vi xử lý 32bit khả dụng đơn giản nhất trên thế giới, với chỉ gồm 30.000 transistor (so với bộ vi xử lý lâu hơn bốn năm của Motorola là 68000 với khoảng 68.000 transistor). Sự đơn giản như vậy có được nhờ ARM không có vi chương trình (mà chiếm khoảng 1/4 đến 1/3 trong 68000) và cũng giống như hầu hết các CPU vào thời đó, không hề chứa cache. Sự đơn giản này đưa đến đặc điểm tiêu thụ công suất thấp của ARM. Thế hệ sau ARM3 được tạo ra với 4KB cache và có chức năng được cải thiện tốt hơn nữa.

Vào những năm cuối thập niên 80, hãng máy tính Apple Computer bắt đầu hợp tác với Acorn để phát triển các thế hệ lõi ARM mới. Công việc này trở nên quan trọng đến nỗi Acorn nâng nhóm thiết kế trở thành một công ty mới gọi là Advanced RISC Machines. Vì lý do đó bạn thường được giải thích ARM là chữ viết tắt của Advanced RISC Machines thay vì Acorn RISC Machine. Advanced RISC Machines trở thành công ty ARM Limited khi công ty này được đưa ra sàn chứng khoán London và NASDAQ năm 1998.

Kết quả sự hợp tác này là ARM6. Mẫu đầu tiên được công bố vào năm 1991 và Apple đã sử dụng bộ vi xử lý ARM 610 dựa trên ARM6 làm cơ sở cho PDA hiệu Apple Newton. Vào năm 1994, Acorn dùng ARM 610 làm CPU trong các máy vi tính RiscPC của họ.

Trải qua nhiều thế hệ nhưng lõi ARM gần như không thay đổi kích thước. ARM2 có 30.000 transistors trong khi ARM6 chỉ tăng lên đến 35.000. Ý tưởng của nhà sản xuất lõi ARM là sao cho người sử dụng có thể ghép lõi ARM với một số bộ phận tùy chọn nào đó để tạo ra một CPU hoàn chỉnh, một loại CPU mà có thể tạo ra trên những nhà máy sản xuất bán dẫn cũ và vẫn tiếp tục tạo ra được sản phẩm với nhiều tính năng mà giá thành vẫn thấp.

Thế hệ thành công nhất có lẽ là ARM7TDMI với hàng trăm triệu lõi được sử dụng trong các máy điện thoại di động, hệ thống video game cầm tay, và Sega Dreamcast. Trong khi công ty ARM chỉ tập trung vào việc bán lõi IP, cũng có một số giấy phép tạo ra bộ vi điều khiển dựa trên lõi này.

Dreamcast đưa ra bộ vi xử lý SH4 mà chỉ mượn một số ý tưởng từ ARM (tiêu tán công suất thấp, tập lệnh gọn ...) nhưng phần còn lại thì khác với ARM. Dreamcast cũng tạo ra một chip xử lý âm thanh được thiết kế bởi Yamaha với lõi ARM7. Bên cạnh đó, Gameboy Advance của Nintendo, dùng ARM7 TDMI ở tần số 16,78 MHz.

Hãng DEC cũng bán giấy phép về lõi cấu trúc ARM (đôi khi chúng ta có thể bị nhầm lẫn vì họ cũng sản xuất ra DEC Alpha) và sản xuất ra thế hệ Strong ARM. Hoạt động ở tần số 233 MHz mà CPU này chỉ tiêu tốn khoảng 1 watt công suất (những đời sau còn tiêu tốn ít công suất hơn nữa). Sau những kiện tụng, Intel cũng được chấp nhận sản xuất ARM và Intel đã nắm lấy cơ hội này để bổ sung vào thế hệ già cỗi i960 của họ bằng Strong ARM. Từ đó, Intel đã phát triển cho chính họ một sản phẩm chức năng cao gọi tên là Xscale.

Bảng 2.1: Các họ vi xử lý ARM

Họ	Lõi	Tốc độ xử lý tối đa (MHz)	Kiến trúc
ARM7TDMI	ARM7TDMIS	16.8 MHz	V4T
	ARM710T	40 MHz	
	ARM720T	59.8 MHz	
	ARM740T		
	ARM7EJS		
ARM9TDMI	ARM9TDMI		V4T
	ARM920T	180 MHz	
	ARM922T		
	ARM940T		
ARM9E	ARM946ES		V5TE Và V5TEJ
	ARM966ES		
	ARM968ES		
	ARM926EJS	180 MHz	
	ARM996HS		
ARM10E	ARM1020E		V5TE Và V5TEJ
	ARM1022E		
	ARM1026EJS		
ARM11	ARM1136J(F)S		V6
	ARM1156T2(F)S		V6T2
	ARM1176JZ(F)S		V6Z
	ARM11 MPCore		V6
Cortex	CortexA8	1 GHz	V7A
	CortexR4	600 MHz	V7M
	CortexM3	100MHz	V7R
XScale	80200/IOP310/IOP315		

	80219		
	IOP321		
	IOP33x		
	PXA210/PXA250		
	PXA255	400 MHz	
	PXA26x		
	PXA27x	624 MHz	
	PXA800(E)F		
	Monahans	1.25 GHz	
	PXA900		
	IXC1100		
	IXP2400/IXP2800		
	IXP2850		
	IXP2325/IXP2350		
	IXP42x		
	IXP460/IXP465		

Để đạt được một thiết kế gọn, đơn giản và nhanh, các nhà thiết kế ARM xây dựng nó theo kiểu nổi cứng không có vi chương trình, giống với bộ vi xử lý 8bit 6502 đã từng được dùng trong các máy vi tính trước đó của hãng Acorn.

Cấu trúc ARM bao gồm các đặc tính của RISC như sau:

- Cấu trúc nạp/lưu trữ.
- Không cho phép truy xuất bộ nhớ không thẳng hàng (bây giờ đã cho phép trong lõi Arm v6).
- Tập lệnh trực giao.
- File thanh ghi lớn gồm $16 \times 32\text{bit}$.
- Chiều dài mã máy cố định là 32 bit để dễ giải mã và thực hiện pipeline, để đạt được điều này phải chấp nhận giảm mật độ mã máy.
- Hầu hết các lệnh đều thực hiện trong vòng một chu kỳ đơn.

So với các bộ vi xử lý cùng thời như Intel 80286 và Motorola 68020, trong ARM có một số tính chất khá độc đáo như sau:

- Hầu hết tất cả các lệnh đều cho phép thực thi có điều kiện, điều này làm giảm việc phải viết các tiêu đề rẽ nhánh cũng như bù cho việc không có một bộ dự đoán rẽ nhánh.
- Trong các lệnh số học, để chỉ ra điều kiện thực hiện, người lập trình chỉ cần sửa mã điều kiện.
- Có một thanh ghi dịch đóng thùng 32bit mà có thể sử dụng với chức năng hoàn hảo với hầu hết các lệnh số học và việc tính toán địa chỉ.
- Có các kiểu định địa chỉ theo chỉ số rất mạnh.
- Có hệ thống con thực hiện ngắt hai mức ưu tiên đơn giản nhưng rất nhanh, kèm theo cho phép chuyển từng nhóm thanh ghi.

II. Vi điều khiển AT91SAM9260:

1. Đặc điểm chính:

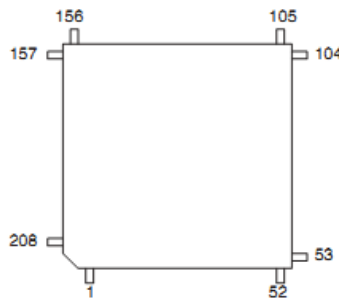
AT91SAM9260 có lõi là bộ vi xử lý ARM926EJS thuộc họ ARM9E hỗ trợ tập lệnh Thumb (tập lệnh nén từ 32 bit xuống 16 bit) và ARM và có thể hoạt động với tốc độ 180Mhz.

- Bộ nhớ:
 - Giao diện bus 32 bit hỗ trợ 4 bank SDRAM, bộ nhớ tĩnh, CompactFlash, NandFlash.
 - Có 2 SRAM được tích hợp bên trong chip hoạt động ở tốc độ cao. Mỗi SRAM có dung lượng 4 Kbyte.
 - Có 1 ROM bên trong chip có dung lượng 32 Kbyte chứa chương trình hệ thống phục vụ cho việc khởi động hệ thống nhúng do nhà sản xuất nạp sẵn,
- Ngoại vi:
 - Bên trong chip tích hợp 4 kênh ADC 10 bit.
 - Có 2 bộ truyền dữ liệu không đồng bộ UART (Universal Asynchronous Receive/Transmitter)
 - Có 4 bộ truyền dữ liệu không đồng bộ/đồng bộ USART (Universal Synchronous Asynchronous Receive/Transmitter).
 - Một giao diện truyền nhận dữ liệu theo chuẩn I2C (chuẩn hai dây).

- Một bộ điều khiển nối tiếp đồng bộ.
- Hai giao tiếp SPI hỗ trợ hai chế độ Master/Slaver.
- Một giao diện SD/MMC (dùng để đọc thẻ nhớ).
- Một bộ điều khiển 10/100 Mbps Ethernet MAC.
- Một bộ điều khiển truyền nhận USB và một bộ truyền nhận USB Device.
- Ba bộ Timer/Counter.
- Hệ thống:
 - Có 22 kênh DMA (Direction Memory Access).
 - Khởi động từ NAND Flash, SDCard, DataFlash hoặc Serial DataFlash.
 - Có bộ điều khiển Reset.
 - Ma trận bus AHB 6 lớp 32 bit hoạt động ở tần số 90Mhz.
 - Một PLL (bộ nhân tần số) cho hệ thống và một cho USB.
 - Hai tín hiệu xung đồng bộ ngoài có thể lập trình được.
 - Có bộ điều khiển ngắt cao cấp và sửa lỗi.
 - Tích hợp bộ dao động nội RC.
 - Cho phép chọn tần số tiết kiệm năng lượng 32,768 Khz và bộ dao động chính 3 đến 20 Mhz.
 - Cho phép cài đặt chế độ timer, watchdog timer, realtime timer.
 - Khối xuất nhập:
 - Ba bộ điều khiển Input/Output song song 32 bit.
 - Có 96 đường Input/Output đa mục đích.

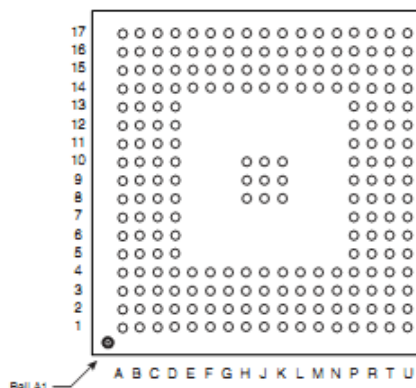
2. Sơ đồ chân của AT91SAM9260:

- AT91SAM9260 có bề ngoài hình vuông, có hai dạng sơ đồ chân: một dạng 208 chân và một dạng 217 chân.
- Dạng 208 chân: Các chân được đưa ra xung quanh của gói nhựa bọc lõi và được đặt tên theo số từ 1 đến 208. Hình dạng được minh họa như sau:



Hình 2.1: Sơ đồ chân của AT91SAM9260 208 chân

- Dạng 217 chân: các chân được đưa ra ở phía dưới của gói nhựa bọc lõi. Hình dạng được minh họa như sau:



Hình 2.2: Sơ đồ chân của AT91SAM9260 217 chân

Vị trí của các chân này được xác định bằng cách ghép các chữ cái của hàng với các số của cột và chân gần với chấm đen làm dấu trên thân của chip là chân A1.

Vì số lượng chân của chip rất lớn nên trong giáo trình này không trình bày cụ thể chức năng của từng chân. Nếu người học muốn tìm hiểu thêm về sơ đồ chân của AT92SAM9260 thì có thể tham khảo Datasheet của AT91SAM9260.

3. Nguồn cấp cho AT91SAM9260:

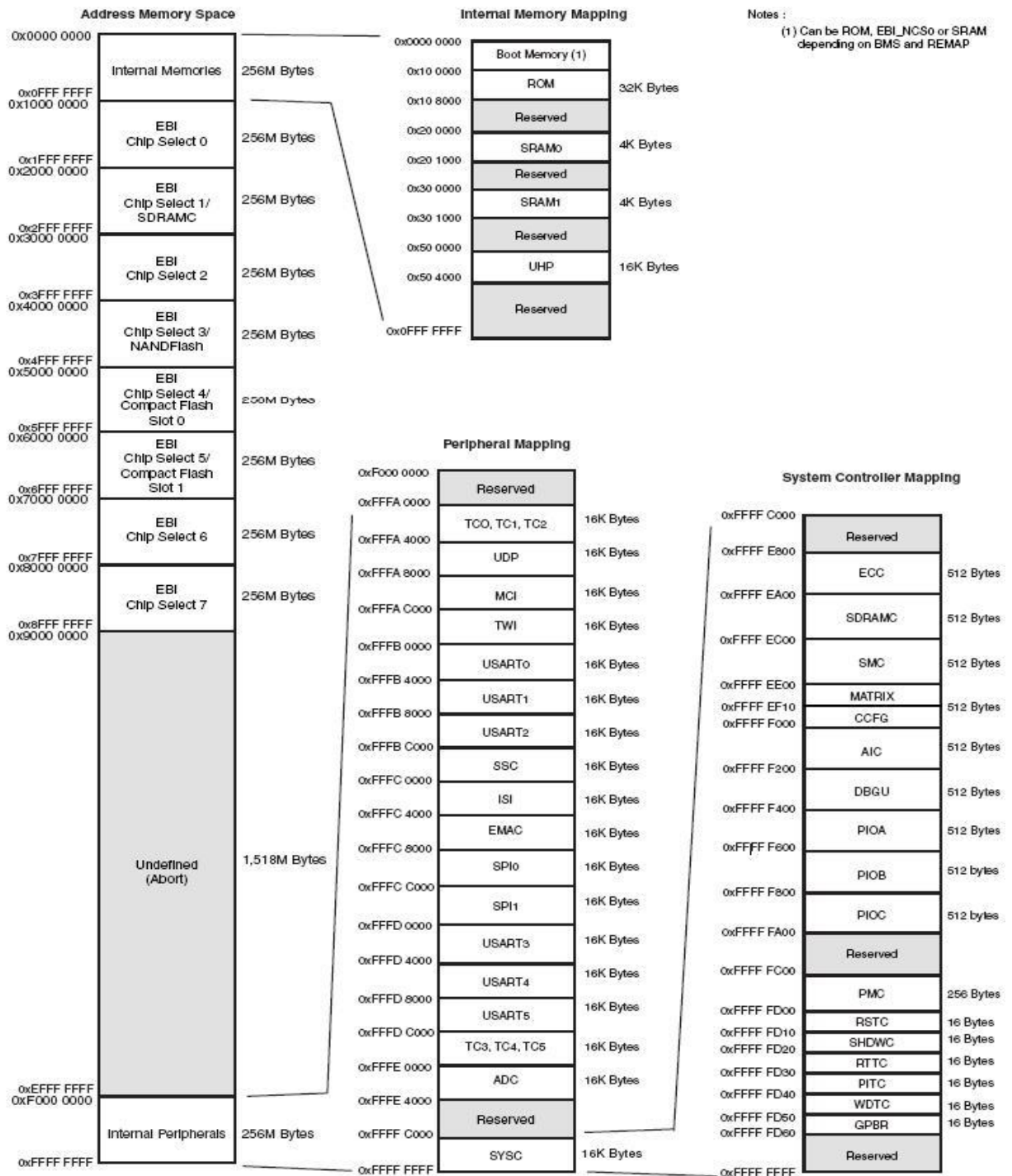
Đặc tính tiết kiệm năng lượng là một trong những thế mạnh của họ vi xử lý ARM. Để tiết kiệm được năng lượng, bên cạnh một kiến trúc lõi tiết kiệm năng lượng thì vi điều khiển còn được thiết kế có nhiều nguồn cấp với các mức điện áp khác nhau.

AT91SAM9260 cần có hai nguồn cấp với các mức điện áp 1,8V và 3,3V. Hai nguồn cấp này được cung cấp tới các chân nguồn của chip để có thể cấp nguồn cho từng modul tích hợp trong chip. Cụ thể như sau:

- Chân VDDCORE: chân này cấp nguồn để nuôi lõi vi xử lý với điện áp là 1,8V.
- Chân VDDIOM: cung cấp nguồn cho bộ giao tiếp Input/Output mở rộng. Điện áp cấp đến chân này là 1,8 V hoặc 3,3 V được chọn bởi phần mềm.
- Chân VDDIOP0: cung cấp cho bộ giao tiếp Input/Output ngoại vi (USB, Micro SD) với điện áp 3,3V.
- Chân VDDDBU: cung cấp điện áp cho bộ dao động chậm (dao động nội RC và thạch anh 32,768 Khz) với điện áp 1,8V.
- Chân VDDPLL: cung cấp điện áp cho bộ dao động chính và bộ nhân tần số với điện áp là 1,8V.
- Chân VDDANA: cung cấp điện áp cho bộ ADC với điện áp 3,3V.

4. Bảng đồ vùng nhớ của AT91SAM9260:

AT91SAM9260 có thể quản lý một vùng nhớ lên đến 4Gbyte địa chỉ, vùng nhớ này được phân thành 16 bank với mỗi bank là 256Mbyte địa chỉ. Mỗi bank này sẽ quản lý bộ nhớ của một thiết bị nhớ nhất định hoặc một vùng nhớ nhất định. Mặc dù thiết bị nhớ hoặc vùng nhớ không chứa dữ liệu hoặc không đượg dùng trong hệ thống, nhưng AT91SAM9260 vẫn để dành vùng nhớ đó cho thiết bị nhớ hoặc vùng nhớ của nó. Cụ thể sẽ được trình bày trong hình sau:



Hình 2.3: Bảng đồ quản lý vùng nhớ của AT91SAM9260

Theo hình trên thì chúng ta có thể thấy bảng đồ vùng nhớ của AT91SAM9260 được chia ra thành nhiều bank để quản lý. Bank trên cùng là bank 0, tiếp theo là bank 1, cho đến cuối cùng là bank 15. Các địa chỉ nằm ở bên trái các cột chính là các địa chỉ vật lý. Bên phải các cột là dung lượng của vùng nhớ tương ứng.

Chúng ta có thể thấy rằng mỗi thiết bị nhớ hoặc vùng nhớ đều được vi xử lý quản lý trong một khoảng địa chỉ nhất định. Nếu thiết bị nhớ được quản lý bởi bank 1 thì ô nhớ đầu tiên của thiết bị nhớ đó (có địa chỉ là 0x00000000) sẽ được vi xử lý đọc với địa chỉ là 0x10000000, ô nhớ thứ hai (có địa chỉ 0x00000001) sẽ được đọc với địa chỉ là 0x10000001. Tương tự thì ô nhớ cuối cùng của thiết bị nhớ đó (có địa chỉ 0x0FFFFFFF) được vi xử lý đọc với địa chỉ là 0x1FFFFFFF. Như vậy, chúng ta có thể thấy địa chỉ thực tế của ô nhớ được vi xử lý đọc với một địa chỉ hoàn toàn khác. Để thuận tiện cho việc tìm hiểu sau này và tránh nhầm lẫn về địa chỉ chúng ta sẽ tìm hiểu về các loại địa chỉ:

Địa chỉ vật lý: là địa chỉ mà vi xử lý gán cho vùng nhớ cần được quản lý. Theo hình trên thì địa chỉ vật lý chính là các địa chỉ nằm ở bên trái các cột.

Địa chỉ đoạn: là địa chỉ vật lý tại ô nhớ đầu tiên của vùng nhớ. Theo hình trên thì địa chỉ đoạn của bank 1 là 0x10000000.

Địa chỉ offset: là khoảng cách tính từ ô nhớ đang xét đến địa chỉ đoạn. Như vậy địa chỉ offset có thể hiểu là địa chỉ thực tế của thiết bị nhớ. Địa chỉ offset của ô nhớ thứ nhất của thiết bị nhớ là 0x00000000, của ô nhớ thứ hai là 0x00000001 và của ô nhớ cuối cùng là 0x0FFFFFFF.

Bank 0 dùng để quản lý bộ nhớ nội của của vi xử lý. ROM được quản lý với địa chỉ vật lý từ 0x00100000 đến 0x00108000 với dung lượng là 32Kbyte. Tương tự vùng nhớ SRAM0 cũng được quản lý từ 0x00200000 đến 0x00201000 ..., các vùng nhớ reserved là các vùng nhớ dự trữ.

Từ Bank 1 đến bank 8 dùng để quản lý các thiết bị nhớ bên ngoài. Cụ thể, bank 2 quản lý SDRAMC với chân chọn chip là NCS1. Bank 4 quản lý NandFlash với chân chọn chip là NCS3/NANDCS. Khi vi xử lý truy cập các thiết bị nhớ này thì vi xử lý sẽ đưa tín hiệu chọn vùng nhớ ra các chân NCS0 đến NCS7 chọn chip tương ứng.

Bank 15 quản lý vùng nhớ của các thiết bị ngoại vi.

Các vùng nhớ có màu đậm là các vùng nhớ dự trữ hoặc là chưa sử dụng đến.

III. Kết luận:

AT91SAM9260 là một vi xử lý có tốc độ hoạt động cao 180 Mhz, nhiều ngoại vi, tiết kiệm năng lượng. Vi điều khiển này sử dụng hai nguồn điện áp là 1,8V và 3,3V cho các modul khác nhau. Ngoài ra vi điều khiển này có thể quản lý một vùng nhớ lên đến 4Gbyte và được chia thành nhiều vùng quản lý ứng với các vùng nhớ khác nhau.

Như vậy, sau khi đọc xong chương này người học đã có một nền kiến thức về vi điều khiển AT91SAM9260. Các kiến thức này sẽ hỗ trợ cho quá trình lập trình ứng dụng sau này của người học.

BÀI 2

**MỘT SỐ THAO TÁC CƠ BẢN
TRÊN HỆ ĐIỀU HÀNH LINUX.**

A-Tổng quan về Linux:

Hệ điều hành Linux là một hệ điều hành mã nguồn mở và còn khá non trẻ so với hệ điều hành Window. Nhưng với những ưu điểm của mình: miễn phí, linh hoạt, uyển chuyển, độ an toàn cao, thống nhất trên các hệ thống phần cứng, mã nguồn mở... Linux đã phát triển mạnh mẽ, đa dạng và chiếm được sự tin tưởng của các lập trình viên.

Các hệ thống nhúng hiện nay đa số chạy trên hệ điều hành Linux. Vì vậy muốn ứng dụng được hệ thống nhúng thì điều tiên quyết là người dùng phải biết sử dụng hệ điều hành Linux.

Phần này sẽ trình bày về các khái niệm về phân vùng đĩa, cách truy xuất ổ đĩa trong Linux, giới thiệu các thư mục hệ thống, màn hình terminal, tập lệnh cơ bản của hệ điều hành Linux. Đây là các kiến thức cơ bản đầu tiên người học cần có để bước vào thế giới kỳ diệu của Linux.

I. Quản lý bộ nhớ trong Linux:

1. Phân vùng đĩa:

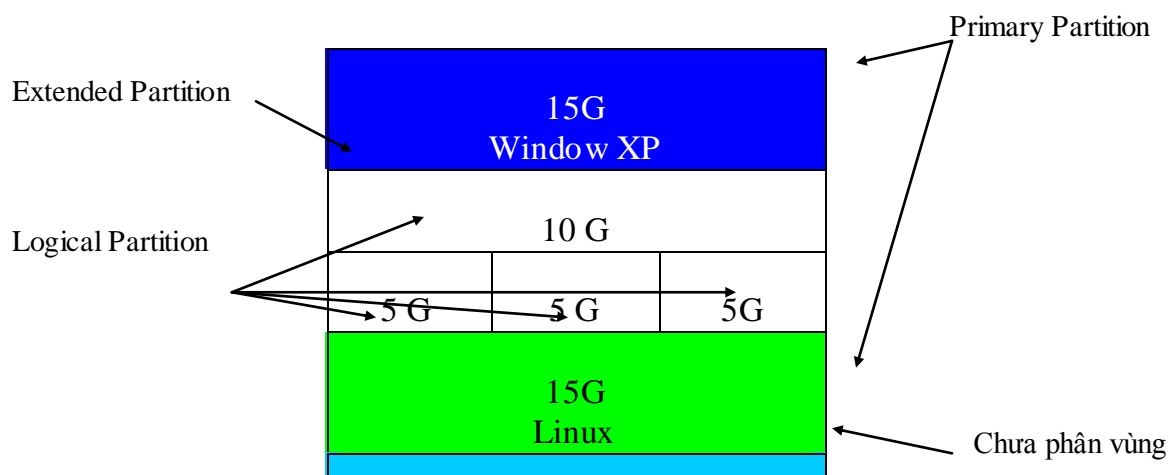
Trong các máy vi tính hiện nay, đĩa cứng là thiết bị lưu trữ được sử dụng phổ biến nhất. Đĩa cứng là một loại đĩa từ, ra đời sau đĩa mềm, có dung lượng lớn 32GB, 80GB và với kỹ thuật tiên tiến như hiện nay thì dung lượng của đĩa cứng đã đạt đến đơn vị TB.

Vì dung lượng của đĩa cứng lớn nên để thuận lợi trong việc quản lý nội dung lưu trên đĩa, đồng thời để tăng hiệu suất sử dụng đĩa cứng thì đĩa cứng sẽ được chia thành nhiều phân vùng hay còn gọi là Partition. Có hai loại Partition là Primary Partition và Extended Partition.

Primary Partition là phân vùng chính, phân vùng này thường được dùng để cài hệ điều hành. Theo nguyên tắc thì trên một đĩa cứng ta chia được tối đa là 4 Primary Partition. Như vậy với một đĩa cứng thì ta chỉ có thể cài tối đa là 4 hệ điều hành.

Lưu ý rằng : ta không thể chia Primary Partition thành các phân vùng nhỏ hơn và Primary Partition ngoài việc dùng để cài đặt hệ điều hành thì ta có thể dùng Primary Partition để lưu trữ dữ liệu.

Extended Partition là phân vùng mở rộng. Thực ra Extended Partition giống như Primary Partition, nhưng với Extended Partition ta có thể chia thành nhiều phân vùng nhỏ hơn bên trong Extended Partition và được gọi là Logical Partition (Phân vùng logic). Extended Partition thường dùng để lưu trữ dữ liệu của người dùng.



Hình 2.4 : Phân vùng của đĩa cứng

Hình trên minh họa cho việc phân vùng các Partition trên một đĩa cứng 60G.

- Theo hình trên, phân vùng màu xanh dương 15G và màu xanh lá cây 15G là các Primary Partition. Hai phân vùng này chứa hệ điều hành. Mỗi Primary Partition chỉ được chứa một hệ điều hành, hai hệ điều hành có thể tồn tại trên cùng một đĩa cứng nếu chúng được cài đặt trên hai Primary Partition khác nhau và dung lượng đĩa cho phép.
- Phân vùng màu trắng 25G là Extended Partition thường được dùng để lưu trữ dữ liệu của người dùng. Phân vùng này được chia thành 4 phân vùng nhỏ khác là các Logical Partition có dung lượng lần lượt là 10G, 5G, 5G, 5G.
- Phân vùng màu xanh lơ là vùng đĩa chưa được phân vùng. Thông thường đối với các phân vùng này thì hệ thống mặc định là Extended Partition.
- Như vậy đối với đĩa cứng trên thì ta có 2 Primary Partition, 2 Extended Partition, 4 Logical Partition.

Mặc dù ta đã chia đĩa cứng thành các phân vùng, nhưng khi cài hệ điều hành lên đĩa cứng thì mỗi hệ điều hành lại nhận diện và truy xuất các phân vùng này theo cách khác nhau. Ví dụ : hệ điều hành Window thì nhận diện các phân vùng này là các ổ đĩa (đĩa C là

Primary Partition, đĩa D,E,... là các Logical Partition) và truy xuất phân vùng này thông qua các ổ đĩa này. Còn đối với hệ điều hành Linux thì nhận diện các phân vùng này là các tập tin và việc truy xuất các phân vùng này cũng thông qua các tập tin này.

Kit KM9260 sử dụng hệ điều hành Linux, nên trong giáo trình này chỉ đề cập đến cách nhận diện và truy xuất phân vùng trên đĩa cứng của Linux.

2. Phân vùng của Linux :

Như đã nói ở trên, Linux nhận diện tất cả các loại đĩa và thiết bị là các tập tin nên việc truy xuất các đĩa và thiết bị cũng thông qua các tập tin này.

Đối với ổ đĩa mềm và các phân vùng của đĩa cứng Linux quy ước cách đặt tên như sau : ổ đĩa mềm thứ nhất là fd0, ổ đĩa mềm thứ hai là fd1 Đĩa cứng thứ nhất là hda, đĩa cứng thứ hai là hdb Nếu có dùng USB thì là sda, sdb, sdc, Các phân vùng chính (Primary Partition hay Extended Partition) được đánh số từ 1 đến 4. Các phân vùng Logical Partition được đánh số từ 5 trở lên.

Ví dụ: hda1 là phân vùng chính của đĩa cứng thứ nhất, hda2 là phân vùng mở rộng của đĩa cứng thứ nhất. Còn hda5, hda6, hda7 là các phân vùng Logical Partition trong Extended Partition.

Các tập tin truy xuất phân vùng này thường được Linux lưu trong thư mục /dev. Chúng ta lưu ý đến ký tự “/” trong Linux. Linux quy ước ký hiệu “/” là Primary Partition của hệ thống, là phân vùng chứa nhân hệ điều hành Linux. Tất cả các đường dẫn trong Linux đều phải được bắt đầu bởi ký hiệu này. Như vậy, các tập tin truy xuất phân vùng và ổ đĩa nằm ở trong thư mục /dev có nghĩa là thư mục dev nằm trong Primary Partition, các tập tin fd0, fd1, ..., hda0, hda1 nằm trong thư mục dev nhưng khi truy xuất các tập tin này là ta truy xuất các phân vùng tương ứng trên đĩa cứng chứ không phải chỉ truy xuất Primary Partition. Ký hiệu “/” còn được dùng làm dấu phân cách thư mục. Ví dụ: ta có đường dẫn: /home/Nhan/Kernel. Theo đường dẫn này ta thấy thư mục home nằm trong thư mục gốc (Primary Partition) và trong thư mục home có thư mục Nhan, trong thư mục Nhan có thư mục Kernel. Để phân cách giữa các thư mục ta dùng ký hiệu “/”.

3. Cách truy xuất ổ đĩa trong Linux:

Trong Linux không có khái niệm ổ đĩa như trong Window. Nên việc truy xuất các phân vùng Linux sử dụng phép gán (mount) tên ổ đĩa với tên một thư mục bất kỳ.

Khi khởi động hệ điều hành, Linux gán cho phân vùng chính (Primary Partition) ký hiệu “/”, các thông tin của phân vùng khác được Linux đặt vào trong thư mục /dev của phân vùng chính. Nếu muốn truy xuất các phân vùng này thì ta thực hiện thao tác kết gán bằng lệnh mount.

Ví dụ:

Trong thư mục /dev có các tập tin:

fd0: ổ mềm thứ nhất

hda1: Primary Partition

hda5, hda6, hda7: Logical Partition thứ nhất, thứ hai, thứ 3 nằm trong Extended Partition.

Trong thư mục /home có các thư mục:

diamem

odiachinh

odiaphu1

odiaphu2

odiaphu3

Muốn truy xuất các phân vùng và ổ đĩa thì ta phải dùng lệnh sau:

```
mount /dev/fd0 /home/diamem
```

```
mount /dev/hda1 /home/odiachinh
```

```
mount /dev/hda5 /home/odiaphu1
```

```
mount /dev/hda6 /home/odiaphu2
```

```
mount /dev/hda7 /home/odiaphu3
```

Lúc này các thao tác sao chép, cắt, dán file vào trong các thư mục diamem, odiachinh, odiaphu1, odiaphu2, odiaphu3 cũng tương đương với việc truy xuất dữ liệu tương ứng vào đĩa mềm, Primary Partition và các Logical Partition.

Khi không muốn sử dụng phân vùng nữa ta có thể dùng lệnh umount để tháo kết gán:

```
umount diamem
```

```
umount odiachinh
```

```
umount odiaphu1
```

```
umount odiaphu2
```

umount odiaphu3

4. Các thư mục trên Linux:

Khi cài đặt hệ điều hành xong thì ta sẽ thấy trong Primary Partition có rất nhiều thư mục. Các thư mục này có các chức năng và mục đích sử dụng nhất định. Cơ bản một hệ thống Linux thường có các thư mục sau:

/bin: Thư mục này chứa các file chương trình thực thi (dạng nhị phân) và file khởi động của hệ thống.

/boot: Các file ảnh (image file) của kernel dùng cho quá trình khởi động thường đặt trong thư mục này.

/dev: Thư mục này chứa các file thiết bị. Trong thế giới Linux các thiết bị phần cứng được xem như là các file.

/ect: Thư mục này chứa các file cấu hình toàn cục của hệ thống. Có thể có nhiều thư mục con trong thư mục này nhưng nhìn chung chúng chứa các file script để khởi động hay phục vụ cho mục đích cấu hình chương trình trước khi chạy.

/home: Thư mục này chứa các thư mục con đại diện cho mỗi user khi đăng nhập. Nơi đây tựa như ngôi nhà của người dùng. Khi người quản trị tạo tài khoản cho bạn, họ cấp cho bạn một thư mục con trong */home*. Bạn hoàn toàn có quyền sao chép, xóa file, tạo thư mục con trong thư mục home của mình mà không ảnh hưởng đến các người dùng khác.

/lib: Thư mục này chứa các file thư viện .so hoặc .sa. Các thư viện C và các thư viện liên kết động cần cho chương trình khi chạy và cho toàn hệ thống.

/lostfound: Khi hệ thống khởi động hoặc khi chạy chương trình fsck, nếu tìm thấy một chuỗi dữ liệu nào đó bị thất lạc trên đĩa cứng không liên quan đến tập tin, Linux sẽ gộp chúng lại và đặt trong thư mục này để nếu cần bạn có thể đọc và giữ lại dữ liệu bị mất.

/mnt: Thư mục này chứa các thư mục kết gán tạm thời đến các ổ đĩa hay thiết bị khác. Bạn có thể thấy trong */mnt* các thư mục con như *cdrom* hoặc *floppy*.

/sbin: Thư mục này chứa các file hay chương trình thực thi của hệ thống thường chỉ cho phép sử dụng bởi người quản trị.

/tmp: Thư mục này chứa các file tạm mà chương trình sử dụng chỉ trong quá trình chạy. Các file trong thư mục này sẽ được hệ thống dọn dẹp nếu không cần dùng đến nữa.

/usr: Thư mục này chứa rất nhiều thư mục con như */usr/bin* hay */usr/sbin*. Một trong những thư mục con quan trọng trong */usr* là */usr/local*, bên trong thư mục *local* này bạn có đủ các thư mục con tương tự ngoài thư mục gốc như *sbin*, *lib*, *bin*, Nếu bạn nâng cấp hệ thống thì các chương trình bạn cài đặt trong *usr/local* vẫn giữ nguyên và bạn không sợ chương trình bị mất mát. Hầu hết các ứng dụng Linux đều thích cài chương trình vào trong */usr/local*.

/var: Thư mục này chứa các file biến thiên bất thường như các file dữ liệu đột nhiên tăng kích thước trong một thời gian ngắn sau đó lại giảm kích thước xuống còn rất nhỏ. Điển hình là các file dùng làm hàm đợi chứa dữ liệu cần đưa ra máy in hoặc các hàng đợi chứa mail.

/usr/include hoặc */usr/local/include*: chứa các file khai báo hàm (header) cần dùng khi biên dịch chương trình nguồn sử dụng các thư viện của hệ thống.

/usr/src: Chứa mã nguồn.

/usr/man: Chứa tài liệu hướng dẫn.

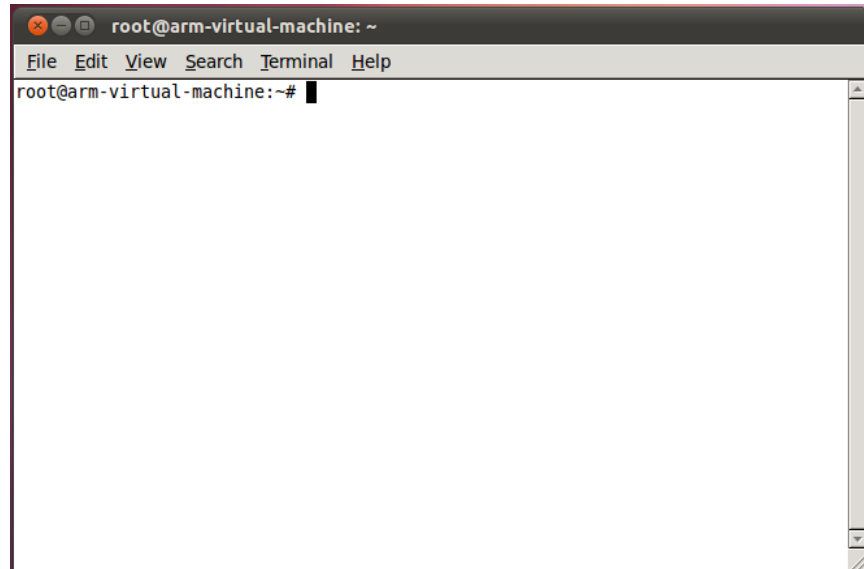
II. Màn hình terminal:

Ngày nay với sự phát triển hiện đại của công nghệ sản xuất chip và sự tiến bộ trong việc lập trình hệ điều hành thì chúng ta đã rất là quen thuộc với các hệ điều hành có hỗ trợ đồ họa như Window XP, Window 7, Linux Ubutu. Với chức năng hỗ trợ đồ họa của mình, các hệ điều hành này giúp cho người sử dụng thuận tiện trong việc thực hiện các thao tác (chỉ cần dùng chuột để thực hiện lệnh) và tạo ra giao diện tương tác thân thiện đối với người dùng. Người dùng không cần phải biết tập lệnh của hệ điều hành mà chỉ cần biết thao tác thực hiện. Ví dụ: để sao chép một tập tin trong Ubutu, người dùng không cần phải biết lệnh sao chép của hệ điều hành là gì, mà chỉ cần biết thao tác sao chép file là được: click chuột phải vào file, chọn copy, sau đó đến vùng cần copy tới, click chuột phải và chọn past into folder.

Tuy nhiên trong thực tế không phải hệ điều hành nào cũng có hỗ trợ đồ họa và không phải hệ thống nào nhúng nào cũng có hỗ trợ đồ họa. Vì vậy mà hầu hết các hệ điều hành hiện nay vẫn còn giữ phương thức giao tiếp với người sử dụng thông qua các dòng

lệnh. Tồn tại song song với phương thức giao tiếp qua đồ họa, phương thức giao tiếp thông qua các dòng lệnh phát huy tác dụng trong các hệ thống nhúng như KIT KM9260, các trường hợp sự cố mà chức năng đồ họa không thể hoạt động được.

Trong Linux Ubutu hay các phiên bản Linux có hỗ trợ đồ họa thì phương thức giao tiếp thông qua các dòng lệnh được thực hiện trên màn hình Terminal



Hình 2.5: Màn hình Terminal của Linux

KIT KM9260 sử dụng hệ điều hành Linux không có hỗ trợ đồ họa, vì vậy mà việc giao tiếp thông qua các dòng lệnh là hết sức quan trọng. Khi khởi động thì màn hình Terminal của Linux sẽ tự động được kích hoạt.

III. Tập lệnh cơ bản của Linux:

- Nhóm lệnh hệ thống:

Lệnh	Cú pháp	Chức năng
ls	ls <thư mục>	Liệt kê nội dung tập tin thư mục
find	find <tên file>	Tìm file
cp	cp file1 file2	Sao chép file
cat	cat > <tên file> cat <tên file>	Tạo file văn bản mới Hiển thị nội dung file
vi	vi <tên file>	Soạn thảo nội dung file
man	man <từ khóa>	Hướng dẫn về lệnh
mv	mv file1 file2	Đổi tên file, thư mục, di chuyển file
rm	rm file	Xóa file

	<code>rm -r <tên thư mục></code>	Xóa cây thư mục
<code>mkdir</code>	<code>mkdir <tên thư mục></code>	Tạo thư mục mới
<code>rmdir</code>	<code>rmdir <tên thư mục></code>	Xóa thư mục rỗng
<code>clear</code>	<code>clear</code>	Xóa màn hình
<code>cd</code>	<code>cd <đường dẫn></code>	Di chuyển đến thư mục khác
<code>chmod</code>	<code>chmod <tham số><tên file></code>	Đổi thuộc tính file, thư mục
<code>uname</code>	<code>uname -r</code>	Xem phiên bản của hệ điều hành
<code>pwd</code>	<code>pwd</code>	Xem đường dẫn thư mục hiện hành
<code>ps</code>	<code>ps</code>	Xem thông tin về tiến trình
<code>kill</code>	<code>kill <số PID ></code>	Hủy tiến trình
<code>gunzip</code>	<code>gunzip <tên file></code>	Giải nén file
<code>gzip</code>	<code>gzip <tên file></code>	Nén file
<code>tar</code>	<code>tar <tham số> <tên file></code>	Nén và giải nén file
<code>env</code>	<code>env</code>	Xem thông tin về biến môi trường
<code>export</code>	<code>export <tên biến>= <nội dung của biến></code>	Thiết lập biến môi trường
<code>echo</code>	<code>echo <tên biến></code>	In chuỗi ra màn hình
<code>df</code>	<code>df</code>	Xem dung lượng đĩa
<code>fsck</code>	<code>fsck</code>	Kiểm tra đĩa và hệ thống file

Bảng 2.2: Các lệnh hệ thống của Linux

- Nhóm lệnh quản lý tài khoản đăng nhập:

Lệnh	Cú pháp	Chức năng
Useradd	<code>user <tên tài khoản></code>	Thêm tài khoản người dùng
Userdel	<code>userdel <tên tài khoản></code>	Xóa tài khoản người dùng
Groupadd	<code>groupadd <tên nhóm></code>	Tạo nhóm mới
Groupdel	<code>groupdel <tên nhóm></code>	Xóa nhóm

Bảng 2.3: Các lệnh quản lý tài khoản của Linux

Cách sử dụng chi tiết các lệnh này sẽ được trình bày kỹ trong phần tiếp theo là: các thao tác cơ bản trên hệ điều hành Linux.

IV. Kết luận:

Linux quản lý các vùng nhớ bằng các tập tin trong thư mục /dev. Người dùng truy xuất các vùng nhớ này bằng cách kết gán các tập tin này với thư mục bất kỳ trong Linux.

Trong Linux chúng ta thao tác các lệnh thông qua màn hình terminal hoặc chương trình giao diện X-Window. Nhưng thao tác lệnh thông qua màn hình terminal được sử dụng nhiều hơn đối với các hệ thống nhúng không hỗ trợ đồ họa như Kit nhúng KM9260..

Như vậy, sau khi đọc xong phần này người học đã phần nào có khái niệm về hệ điều hành Linux. Các kiến thức này sẽ là nền tảng cho người học trong những phần sau. Trong phần tiếp theo chúng ta sẽ tìm hiểu kỹ hơn về các lệnh trong Linux.

B-Các thao tác cơ bản trong hệ điều hành Linux

Như đã nêu trong phần trước, người dùng thường thao tác với hệ điều hành Linux trên các Kit nhúng thông qua màn hình terminal bằng cách ra lệnh cho hệ điều hành. Vì vậy để dùng được Linux thì chúng ta phải biết được các lệnh trong Linux.

Trong phần này sẽ trình bày các lệnh của Linux hỗ trợ cho các thao tác cơ bản nhất: liệt kê thư mục tập tin, tạo, xóa thư mục tập tin, sao chép thư mục tập tin, di chuyển và đổi tên thư mục tập tin, kết gán ổ đĩa, thay đổi thư mục hiện hành, ... và một số lệnh của hệ thống....

I. Nội dung:

1. Khởi động lại (reset) hệ thống và tắt hệ thống:

- Để khởi động lại hệ thống ta dùng lệnh:

```
$ reset
```

- Để tắt hệ thống ta dùng lệnh:

```
$ poweroff
```

2. Sử dụng tài liệu hướng dẫn man:

Man là một chương trình có sẵn trong Linux có chức năng dò tìm các thông tin mà bạn đăng nhập vào từ dòng lệnh (gọi là khóa hay keyword). Nếu tìm thấy, nội dung chi tiết mô tả về khóa hoặc những thông tin liên quan đến khóa sẽ được hiển thị đầy đủ để bạn tham khảo.

- Bạn có thể gọi trình man theo cú pháp sau:

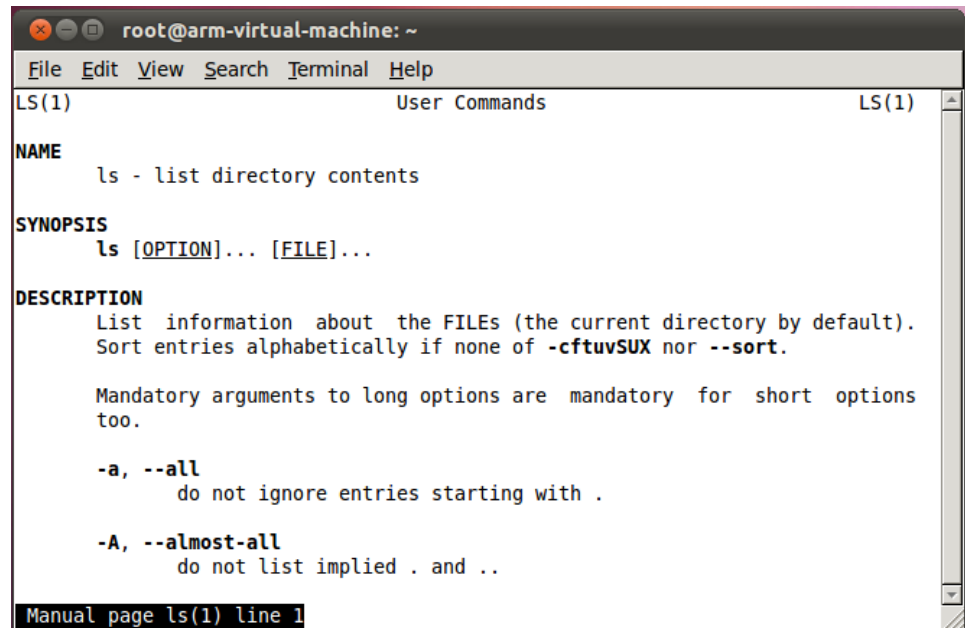
```
/$ man <keyword>
```

- Để thoát khỏi tài liệu man chúng ta có thể gõ q và nhấn Enter

Ví dụ: để tìm hiểu về nội dung và cách sử dụng lệnh ls chúng ta gõ

`/ $ man ls`

Khi đó tài liệu man về lệnh ls sẽ hiện ra



Hình 2.6: Trình man hướng dẫn lệnh ls

- Trong tài liệu man sẽ liệt kê tên, cú pháp, chức năng của lệnh mà ta cần tìm. Ngoài ra, tài liệu man còn liệt kê ra các tham số có thể đi cùng với lệnh.

Ví dụ: như lệnh ls thì ta có thể thấy tài liệu man liệt kê ra hai tham số là `-a` và `-A`. Nếu chúng ta dùng lệnh

`/ $ ls -a`

Thì chương trình sẽ liệt kê tất cả tập tin và thư mục có trong thư mục gốc và hai dấu `.` và `..`. Còn nếu ta dùng lệnh

`/ $ ls -A`

Thì chương trình sẽ liệt kê tất cả các tập tin và thư mục có trong thư mục gốc nhưng sẽ không liệt kê hai dấu `.` và `..`.

- Trình man phân các hướng dẫn ra làm thành từng đoạn (session) với những chủ đề khác nhau gồm:

Mục	Tên đề mục	Nội dung
1	User command	Các lệnh thông thường của hệ điều hành
2	System call	Các hàm kernel của hệ thống
3	Subroutines	Các hàm thư viện
4	Devices	Các hàm truy xuất và xử lý file thiết bị
5	File format	Các hàm định dạng file
6	Games	Các lệnh liên quan đến trò chơi
7	Miscell	Các hàm linh tinh
8	Sys. Admin	Các lệnh quản trị hệ thống

Bảng 2.4: Các nhóm lệnh trình man hỗ trợ

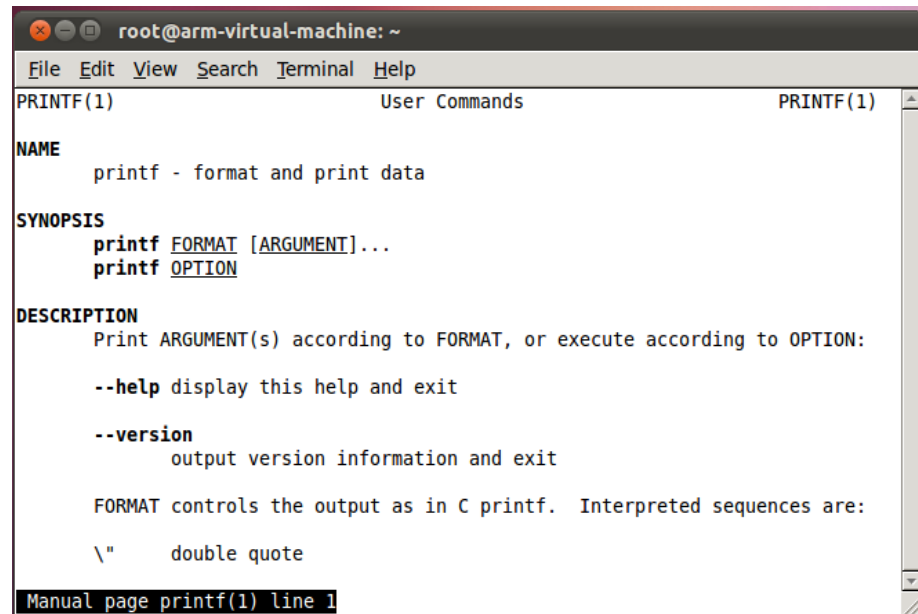
Thông thường nếu chúng ta gọi man và không chỉ định thêm gì cả ngoài từ khóa cần tìm thì man sẽ tìm từ khóa trong phân đoạn 1 (user command). Tuy nhiên chúng ta có thể yêu cầu man tìm trong một phân đoạn khác. Ví dụ từ khóa printf vừa là một lệnh của hệ điều hành vừa là một hàm của C. Nếu chúng ta muốn tìm printf liên quan đến hệ điều hành (user command) chúng ta sẽ gõ:

`/$ man printf`

Hoặc

`/$ man 1 printf`

Khi đó trình man sẽ liệt kê thông tin của lệnh printf liên quan đến hệ điều hành như sau:



```
root@arm-virtual-machine: ~
File Edit View Search Terminal Help
PRINTF(1) User Commands PRINTF(1)

NAME
    printf - format and print data

SYNOPSIS
    printf FORMAT [ARGUMENT]...
    printf OPTION

DESCRIPTION
    Print ARGUMENT(s) according to FORMAT, or execute according to OPTION:

    --help display this help and exit

    --version
        output version information and exit

    FORMAT controls the output as in C printf. Interpreted sequences are:

    \"    double quote

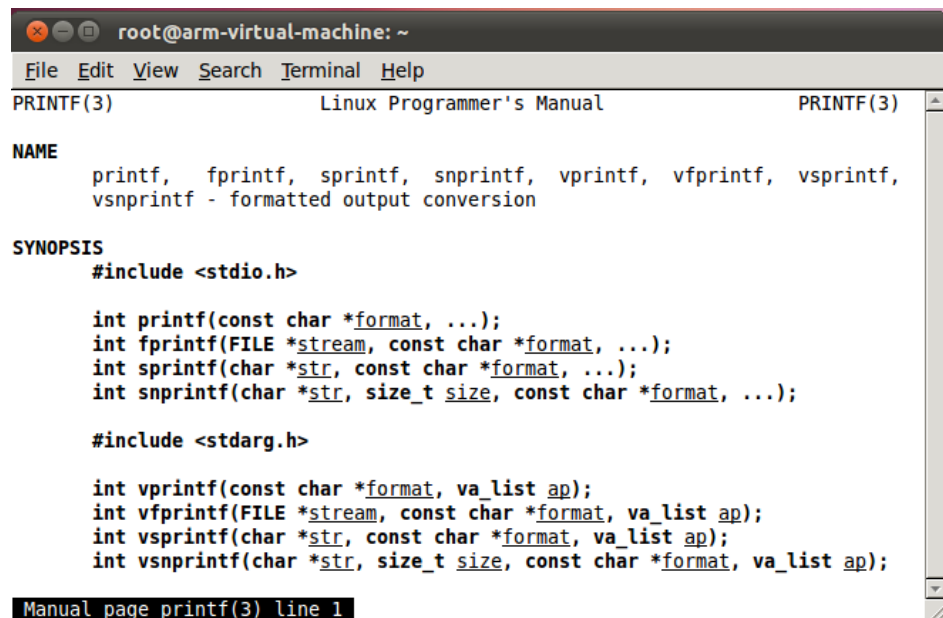
Manual page printf(1) line 1
```

Hình 2.7: Trình man hướng dẫn lệnh printf của hệ điều hành

Còn nếu chúng ta gõ:

/ \$ man 3 printf

Thì man sẽ liệt kê các thông tin của printf liên quan đến hàm thư viện C như sau :



```
root@arm-virtual-machine: ~
File Edit View Search Terminal Help
PRINTF(3) Linux Programmer's Manual PRINTF(3)

NAME
    printf, fprintf, sprintf, snprintf, vprintf, fprintf, vsprintf,
    vsnprintf - formatted output conversion

SYNOPSIS
    #include <stdio.h>

    int printf(const char *format, ...);
    int fprintf(FILE *stream, const char *format, ...);
    int sprintf(char *str, const char *format, ...);
    int snprintf(char *str, size_t size, const char *format, ...);

    #include <stdarg.h>

    int vprintf(const char *format, va_list ap);
    int fprintf(FILE *stream, const char *format, va_list ap);
    int vsprintf(char *str, const char *format, va_list ap);
    int vsnprintf(char *str, size_t size, const char *format, va_list ap);

Manual page printf(3) line 1
```

Hình 2.8: Trình man hướng dẫn lệnh printf trong lệnh C

3. Liệt kê thư mục tập tin:

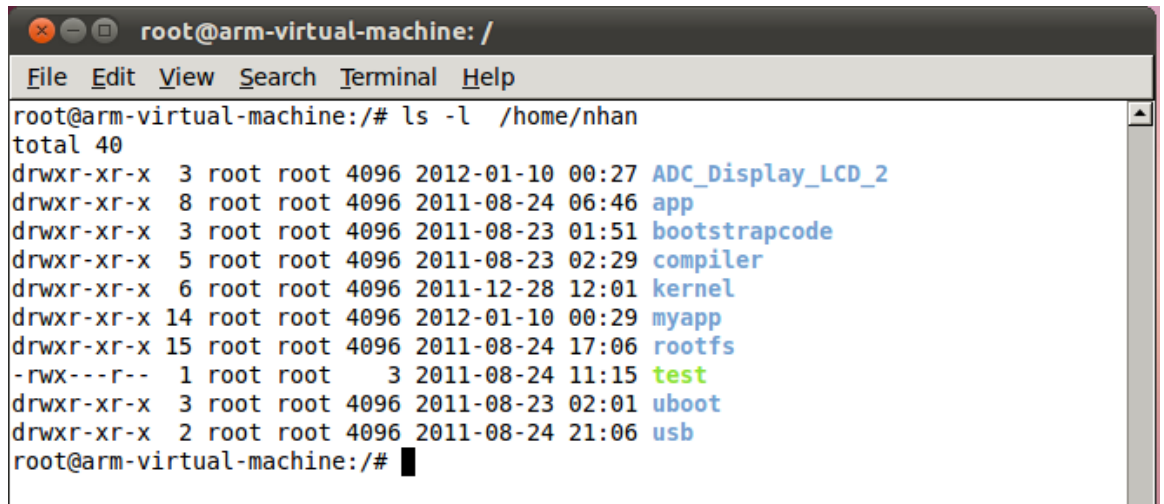
- Để liệt kê thư mục con và tập tin trong một thư mục thì ta dùng lệnh ls. Cú pháp của lệnh ls như sau :

`/ $ ls <đường dẫn và tên thư mục cần liệt kê thư mục con và nội dung>`

- Ngoài ra lệnh ls còn được hỗ trợ thêm tham số :

`/ $ ls -l <đường dẫn và tên thư mục cần liệt kê thư mục con và nội dung>`

(liệt kê các thư mục con và tập tin theo dạng danh sách chi tiết)



```
root@arm-virtual-machine: /  
File Edit View Search Terminal Help  
root@arm-virtual-machine: /# ls -l /home/nhan  
total 40  
drwxr-xr-x  3 root root 4096 2012-01-10 00:27 ADC_Display_LCD_2  
drwxr-xr-x  8 root root 4096 2011-08-24 06:46 app  
drwxr-xr-x  3 root root 4096 2011-08-23 01:51 bootstrapcode  
drwxr-xr-x  5 root root 4096 2011-08-23 02:29 compiler  
drwxr-xr-x  6 root root 4096 2011-12-28 12:01 kernel  
drwxr-xr-x 14 root root 4096 2012-01-10 00:29 myapp  
drwxr-xr-x 15 root root 4096 2011-08-24 17:06 rootfs  
-rwx---r--  1 root root   3 2011-08-24 11:15 test  
drwxr-xr-x  3 root root 4096 2011-08-23 02:01 uboot  
drwxr-xr-x  2 root root 4096 2011-08-24 21:06 usb  
root@arm-virtual-machine: /#
```

Hình 2.9: Lệnh ls liệt kê thư mục tập tin

4. Tạo và xóa thư mục tập tin:

- Để tạo thư mục ta dùng lệnh mkdir

`/ $ mkdir <đường dẫn và tên thư mục cần tạo>`

- Để xóa thư mục thì ta dùng lệnh rmdir

`/ $ rmdir <đường dẫn và tên thư mục cần xóa>`

Ví dụ: đang đứng ở ổ đĩa gốc, tạo thư mục /home/nhan/kernel, sau đó xóa lần lượt thư mục kernel và nhan.

Trước tiên là ta phải tạo thư mục nhan trong thư mục home. Vì thư mục home là thư mục có sẵn trong Linux nên ta không cần phải tạo lại nữa. Sau đó ta tạo thư mục kernel nằm trong thư mục nhan và xóa hai thư mục vừa tạo. Trình tự dòng lệnh như sau :

`/ $ mkdir /home/nhan`

`/ $ mkdir /home/nhan/kernel`

`/ $ rmdir /home/nhan/kernel`

`/ $ rmdir /home/nhan`

- Để tạo ra một tập tin thì ta dùng lệnh `cat`

`/ $ cat > <đường dẫn và tên tập tin muốn tạo>`

- Để xóa một tập tin thì ta dùng lệnh `rm`

`/ $ rm <đường dẫn và tên tập tin muốn xóa>`

- Để xem nội dung một tập tin thì ta dùng

`/ $ cat < đường dẫn và tên tập tin muốn xem>`

Ví dụ : đứng ở thư mục gốc tạo một tập tin `/home/nhan/app/test1`, sau đó xóa tập tin này. Thứ tự dòng lệnh như sau :

`/ $ mkdir /home/nhan`

`/ $ mkdir /home/nhan/app`

`/ $ cat > /home/nhan/app/test1`

Lúc này chúng ta có thể nhập nội dung cho tập tin `test1`. Nếu muốn thoát khỏi tập tin thì ta nhấn `Ctrl+D`

Để xem nội dung của tập tin vừa tạo thì ta dùng

`/ $ cat /home/nhan/app/test1`

Ta xóa tập tin `test1`

`/ $ rm /home/nhan/app/test1`

Lưu ý: trong quá trình tạo thư mục và tập tin thì chúng ta có thể dùng lệnh `ls` để kiểm tra xem tập tin hay thư mục đã được tạo hay chưa.

5. Kết gán ổ đĩa và thư mục:

Như đã nói ở các phần trước, để truy xuất được các thiết bị lưu trữ hoặc các phân vùng thì chúng ta phải kết gán thiết bị lưu trữ hay phân vùng này với file tương ứng trong Linux. Để làm được việc này thì ta dùng lệnh `mount`.

- Lệnh `mount` có cú pháp như sau:

`/ $ mount -t vfstype devicefile mdir`

Trong đó, devicefile là đường dẫn đến file mà Linux nhận diện thiết bị. Tùy chọn -t sẽ kết gán theo kiểu hệ thống file trên thiết bị do vfstype qui định. mdir là đường dẫn kết nối vào hệ thống thư mục của Linux. Hiện nay Linux có thể đọc được rất nhiều hệ thống file, vsftype có thể bao gồm những kiểu hệ thống file thông dụng sau :

Msdos : hệ thống file và thư mục theo bảng FAT16, FAT32 của DOS

Ntfs: định dạng hệ thống file NTFS của Window NT

Ext2: định dạng hệ thống file chuẩn của Linux và Unix

Nfs: định dạng hệ thống file truy xuất qua mạng

Iso9660: hệ thốn file theo chuẩn ISO

- Để tháo kết gán ta dùng lệnh umount:

```
/$ umount mdir
```

Ví dụ1:

Kết gán ổ đĩa A trong Linux để đọc các file theo hệ thống bảng FAT của DOS.

Trước tiên là ta tạo thư mục để kết gán

```
/$ mkdir /dos
```

Gọi lệnh mount để kết gán thư mục

```
/$ mount -t msdos /dev/fd0 /dos
```

Nếu kết gán thành công thì kể từ bây giờ tất cả những gì chúng ta ghi vào thư mục /dos cũng tương đương như chúng ta ghi vào ổ đĩa A.

Để tháo kết gán ta dùng lệnh

```
/$ umount /dos
```

Ví dụ 2 :

Kết gán ổ đĩa CD-ROM vào thư mục /mycdrom

```
/$ mkdir /mycdrom
```

```
/$ mount -t iso9660 /dev/cdrom /mycdrom
```

6. Thay đổi thư mục hiện hành:

- Thư mục hiện hành là thư mục mà con trỏ chương trình đang ở đó. Thư mục hiện hành được xác định bởi thư mục ở trước dấu \$ trên dòng lệnh.

Ví dụ:

- Nếu bắt đầu dòng lệnh là dấu /\$ thư mục hiện hành là thư mục gốc
- Nếu bắt đầu dòng lệnh là /home\$ thì thư mục hiện hành là thư mục /home

- Để thay đổi thư mục hiện hành thì ta dùng lệnh cd. Cú pháp lệnh cd như sau:

```
/$ cd <đường dẫn và tên thư mục muốn làm thư mục hiện hành>
```


Ví dụ:

Ta đang ở thư mục hiện hành là thư mục gốc, ta thay đổi thư mục hiện hành là /home/nhan, giả sử các thư mục đã được tạo sẵn. Ta làm như sau:

```
/$ cd /home/nhan
```

```
/home/nhan$
```

Ta thấy rằng phần bắt đầu dòng lệnh đã thay đổi /\$ thành /home/nhan\$ nghĩa là thư mục hiện hành đã được thay đổi.

Để thay đổi thư mục hiện hành là home thì ta làm như sau:

```
/home/nhan$ cd /home
```

Hoặc

```
/home/nhan$ cd ..
```

7. Sao chép tập tin và thư mục:

- Lệnh cp của Linux dùng để sao chép file. Cú pháp của lệnh cp như sau :

\$ cp <đường dẫn và tên file cần sao chép> <đường dẫn và tên file muốn sao chép đến>

Ví dụ : trong thư mục /home/nhan/app có tập tin test1, chép tập tin test1 sang thư mục /home/nhan/test với tên test2. Giả sử các thư mục đã được tạo trước đó. Thứ tự các dòng lệnh như sau:

```
/$ cp /home/nhan/app/test1 /home/nhan/test/test2
```

- Lệnh cp còn dùng để sao chép cây thư mục. Để sao chép cây thư mục thì ta thêm tùy chọn -R vào lệnh cp.

Ví dụ: sao chép thư mục /home/nhan/app sang thư mục /home ta làm như sau:

```
/$ cp -R /home/nhan/app /home
```

Lúc này trong thư mục /home có thêm thư mục app

8. Di chuyển và đổi tên tập tin, thư mục:

- Chúng ta dùng lệnh mv để di chuyển hoặc đổi tên file. Cú pháp lệnh mv như sau :

/\$ mv <đường dẫn và tên file cần di chuyển hoặc đổi tên> <đường dẫn và tên file mới>

Ví dụ :

▪ Di chuyển file /home/nhan/app/test1 sang thư mục /home. Ta làm như sau:

```
/$ mv /home/nhan/app/test1 /home
```

- Di chuyển file /home/nhan/app/test1 sang thư mục /home với tên test2. Ta làm như sau:

```
/ $ mv /home/nhan/app/test1 /home/test2
```

- Đổi tên file /home/nhan/app/test1 thành /home/nhan/app/test3. Ta làm như sau:

```
/ $ mv /home/nhan/app/test1 /home/nhan/app/test3
```

- Lệnh mv còn dùng để di chuyển thư mục.

Ví dụ:

để di chuyển thư mục /home/nhan/app ra thư mục /home ta làm như sau:

```
/ $ mv /home/nhan/app/ /home
```

9. Phân quyền bảo vệ và truy xuất trên tập tin:

- Một tập tin hay thư mục trong Linux có các thuộc tính sau:

r: chỉ cho đọc

w: cho phép ghi vào tập tin

x: cho phép thực thi chương trình

-: không cho phép

- Đây là các thuộc tính về quyền truy xuất mà hệ điều hành dùng để bảo vệ file và thư mục. Chúng ta có 3 quyền chính trên một file hay thư mục như đã nêu trên. Mặc dù vậy các quyền này được chỉ định cho 3 đối tượng nữa đó là: người sở hữu tập tin (owner), nhóm sở hữu tập tin (group), những người sử dụng tập tin thông thường (other user).

- Tóm lại tập tin của chúng ta được kiểm soát bởi 3 quyền và truy xuất bởi 3 đối tượng khác nhau. Khi chúng ta dùng lệnh

```
/ $ ls -l
```

Thì Linux sẽ liệt kê các thuộc tính của file và thư mục trong thư mục gốc ở đầu các dòng

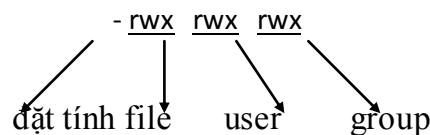
```

root@arm-virtual-machine: /
File Edit View Search Terminal Help
root@arm-virtual-machine:/# ls -l
total 136
drwxr-xr-x  2 root root  4096 2010-12-28 06:02 bin
drwxr-xr-x  3 root root  4096 2010-12-28 06:14 boot
drwxr-xr-x  2 root root  4096 2010-12-23 15:16 cdrom
drwxr-xr-x 17 root root 4440 2011-08-24 03:22 dev
drwxr-xr-x 138 root root 12288 2011-08-24 03:22 etc
-rwxr-xr-x  1 root root  7171 2011-08-23 00:57 helloworld
-rw-r--r--  1 root root    71 2011-08-23 00:57 helloworld.c
drwxr-xr-x  6 root root  4096 2011-08-23 06:29 home
lrwxrwxrwx  1 root root    33 2010-12-28 06:08 initrd.img -> boot/initrd.img
.6.38-10-generic
lrwxrwxrwx  1 root root    33 2010-12-23 15:45 initrd.img.old -> boot/initrd
mg-2.6.35-22-generic
drwxr-xr-x 19 root root 12288 2010-12-28 06:05 lib
drwx----- 2 root root 16384 2010-12-23 15:10 lost+found

```

Hình 2.10: Lệnh ls liệt kê chi tiết thư mục tập tin trong thư mục gốc

- Trên thuộc tính phân quyền của file, đối tượng được chia thành 3 nhóm từ trái sang phải, mỗi nhóm chứa 3 phân quyền đầy đủ như sau:



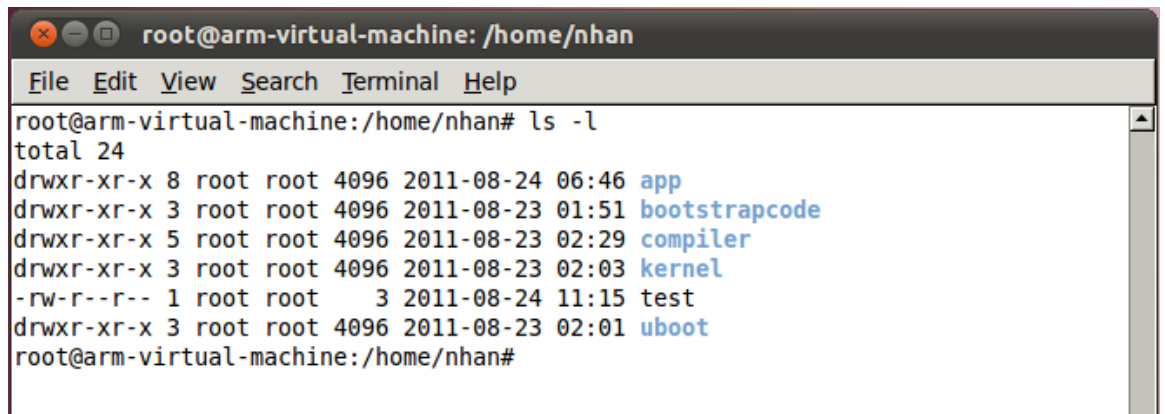
Hình 2.11: Thuộc tính phân quyền của tập tin thư mục

- Cờ đầu tiên cho biết đặt tính của file, thường không quan trọng. Nếu là – có nghĩa là tập tin thông thường. Trường hợp thư mục thì chúng ta sẽ thấy cờ này ký hiệu là d.
- Còn 3 cờ tiếp theo là các thuộc tính tương ứng với 3 nhóm. Nếu tại vị trí của thuộc tính đó mà có dấu – thì có nghĩa là thuộc tính đó không được kích hoạt cho tập tin hay thư mục này. Còn nếu là chữ thì thuộc tính đó được kích hoạt.
- Chúng ta có thể thay đổi việc kích hoạt hay không kích hoạt các thuộc tính này bằng lệnh chmod. Cú pháp của lệnh chmod như sau:

/ \$ chmod <các thuộc tính> <tên file hoặc thư mục>

Ví dụ: ta có thư mục /home/nhan có các file và thư mục được lệnh

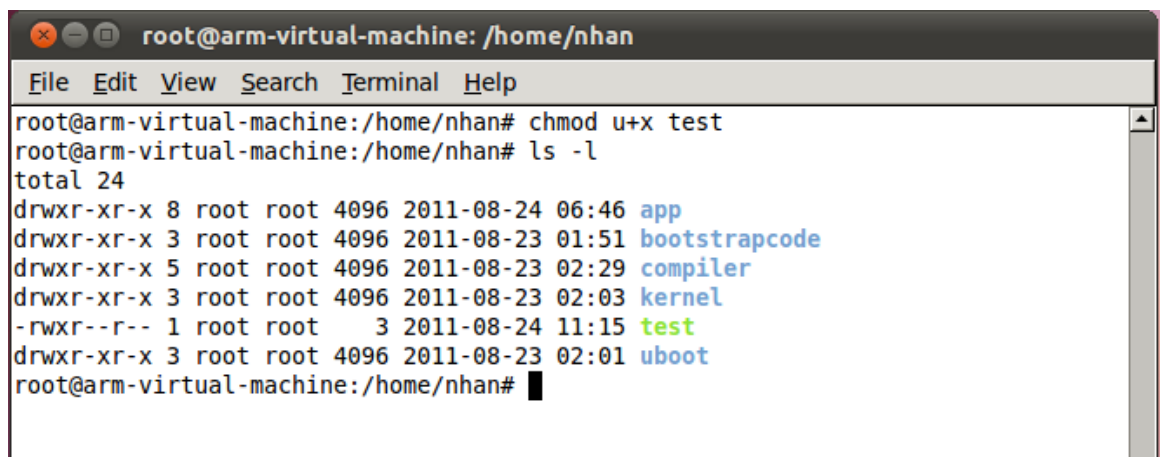
/ \$ ls -l liệt kê như sau:



```
root@arm-virtual-machine: /home/nhan
File Edit View Search Terminal Help
root@arm-virtual-machine:/home/nhan# ls -l
total 24
drwxr-xr-x 8 root root 4096 2011-08-24 06:46 app
drwxr-xr-x 3 root root 4096 2011-08-23 01:51 bootstrapcode
drwxr-xr-x 5 root root 4096 2011-08-23 02:29 compiler
drwxr-xr-x 3 root root 4096 2011-08-23 02:03 kernel
-rw-r--r-- 1 root root 3 2011-08-24 11:15 test
drwxr-xr-x 3 root root 4096 2011-08-23 02:01 uboot
root@arm-virtual-machine:/home/nhan#
```

Hình 2.12: Ví dụ lệnh chmod 1

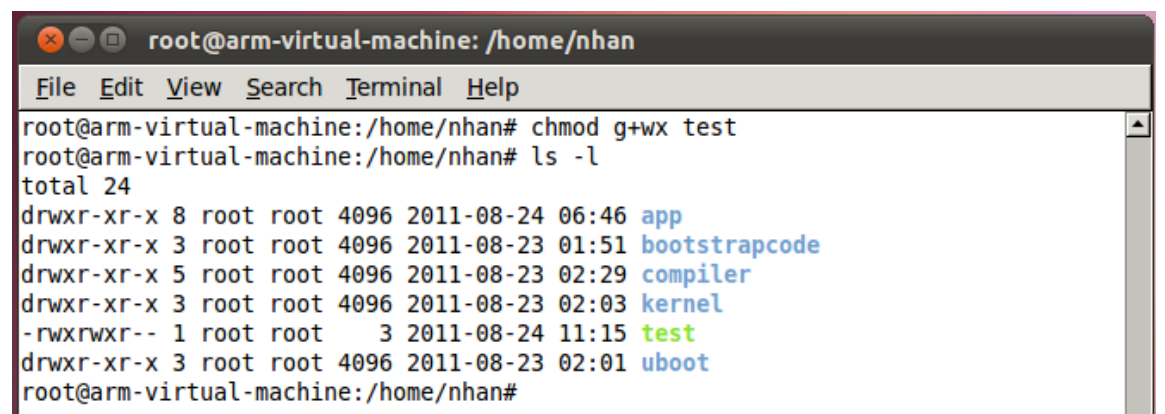
Ta thấy các thư mục sẽ có chữ d ở cờ đầu tiên còn tập tin test thì là dấu -. Tập tin test không có thuộc tính x cho nhóm user, group, other, thuộc tính w cho nhóm group và other. Bây giờ chúng ta sẽ thêm thuộc tính x cho nhóm user như sau:



```
root@arm-virtual-machine: /home/nhan
File Edit View Search Terminal Help
root@arm-virtual-machine:/home/nhan# chmod u+x test
root@arm-virtual-machine:/home/nhan# ls -l
total 24
drwxr-xr-x 8 root root 4096 2011-08-24 06:46 app
drwxr-xr-x 3 root root 4096 2011-08-23 01:51 bootstrapcode
drwxr-xr-x 5 root root 4096 2011-08-23 02:29 compiler
drwxr-xr-x 3 root root 4096 2011-08-23 02:03 kernel
-rwxr--r-- 1 root root 3 2011-08-24 11:15 test
drwxr-xr-x 3 root root 4096 2011-08-23 02:01 uboot
root@arm-virtual-machine:/home/nhan#
```

Hình 2.13: Ví dụ lệnh chmod 2

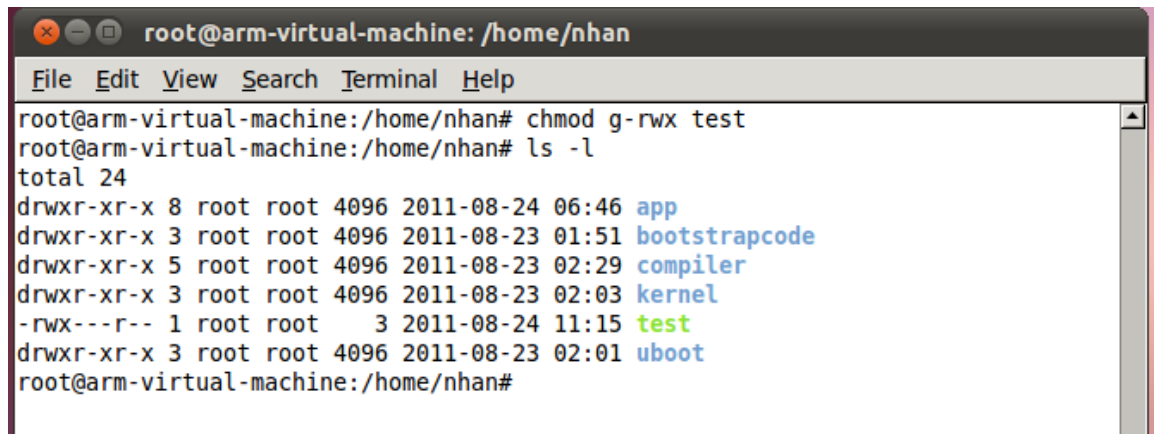
Bây giờ ta thêm thuộc tính w và x cho nhóm group:



```
root@arm-virtual-machine: /home/nhan
File Edit View Search Terminal Help
root@arm-virtual-machine:/home/nhan# chmod g+wx test
root@arm-virtual-machine:/home/nhan# ls -l
total 24
drwxr-xr-x 8 root root 4096 2011-08-24 06:46 app
drwxr-xr-x 3 root root 4096 2011-08-23 01:51 bootstrapcode
drwxr-xr-x 5 root root 4096 2011-08-23 02:29 compiler
drwxr-xr-x 3 root root 4096 2011-08-23 02:03 kernel
-rwxrwxr-- 1 root root 3 2011-08-24 11:15 test
drwxr-xr-x 3 root root 4096 2011-08-23 02:01 uboot
root@arm-virtual-machine:/home/nhan#
```

Hình 2.14: Ví dụ lệnh chmod 3

Ta có thể bỏ kích hoạt tất cả các thuộc tính của group như sau:



```
root@arm-virtual-machine: /home/nhan
File Edit View Search Terminal Help
root@arm-virtual-machine:/home/nhan# chmod g-rwx test
root@arm-virtual-machine:/home/nhan# ls -l
total 24
drwxr-xr-x 8 root root 4096 2011-08-24 06:46 app
drwxr-xr-x 3 root root 4096 2011-08-23 01:51 bootstrapcode
drwxr-xr-x 5 root root 4096 2011-08-23 02:29 compiler
drwxr-xr-x 3 root root 4096 2011-08-23 02:03 kernel
-rwx---r-- 1 root root 3 2011-08-24 11:15 test
drwxr-xr-x 3 root root 4096 2011-08-23 02:01 uboot
root@arm-virtual-machine:/home/nhan#
```

Hình 2.15: Ví dụ lệnh `chmod` 4

10. Nén và giải nén tập tin, thư mục:

- Trong Linux chúng ta có thể nén và giải nén các file như *.gz, *.bz2, *.tar
- Để nén và giải nén một file trong Linux chúng ta dùng lệnh tar.
- Ta dùng lệnh tar để giải nén file *.bz2 như sau :

```
/ $ tar -jxvf *.bz2
```

Trong đó :

tar là tên lệnh

j là tham số dùng khi nén hoặc giải nén file *.bz2

x là tham số dùng khi giải nén một file

v là tham số yêu cầu hiện chi tiết các thông tin trong quá trình nén hay giải nén file

f có tác dụng chỉ định file cần giải nén hay cần nén là *.bz2 ở cuối dòng lệnh.

*.bz2 là file chúng ta cần nén hoặc giải nén.

- Ta dùng lệnh tar để giải nén file *.gz như sau

```
/ $ tar -zxvf *.gz
```

Trong đó:

z là tham số dùng khi nén hoặc giải nén file *.gz

- Ta dùng lệnh tar để giải nén file *.tar như sau:

```
/ $ tar -xvf *.tar
```

- Ta dùng lệnh tar để xem nội dung một file *.bz2, *.gz, *.tar như sau:

```
/ $ tar -tf <tên file>
```

Trong đó:

t là tham số dùng khi ta muốn xem nội dung file nén.

- Ta dùng lệnh tar để nén file *.bz2 như sau:

```
/$ tar -jcvf *.bz2 <file1> <file2>
```

Trong đó :

c là tham số chúng ta dùng khi nén một file.

- Ta dùng lệnh tar để nén file *.gz như sau:

```
/$ tar -zcvf *.gz <file1> <file2>...
```

- Ta dùng lệnh tar để nén file *.tar như sau :

```
/$ tar -cvf *.tar <file1> <file2>...
```

- Ngoài ra, để nén và giải nén file *.gz chúng ta cũng có thể dùng lệnh gzip và lệnh gunzip :

```
/$ gzip <file cần nén>
```

```
/$ gunzip <file *.gz cần giải nén>
```

11. Biên dịch một chương trình ứng dụng:

- Khi chúng ta lập trình một ứng dụng bằng ngôn ngữ C trên Linux thì chúng ta phải biên dịch ra file thực thi ứng dụng đó. Trong Linux chúng ta biên dịch một file ứng dụng như sau :

```
/$ gcc <tên file cần biên dịch> -o <tên file muốn xuất ra>
```

Trong đó :

gcc là tên lệnh dùng để biên dịch một chương trình viết bằng ngôn ngữ C.

-o là tham số yêu cầu chương hệ điều hành tạo ra file thực thi với tên file mà ta muốn được viết ngay sau -o. Nếu chúng ta không dùng tham số này thì hệ điều hành sẽ tự động biên dịch ra file a.out.

12. Cài đặt các thông số cho chuẩn truyền nhận dữ liệu Ethernet:

- Ethernet là chuẩn truyền nhận dữ liệu phổ biến hiện nay. Trong Linux có hỗ trợ các lệnh để người dùng có thể sử dụng chuẩn truyền nhận dữ liệu này.

- Để xem địa chỉ IP của hệ thống ta dùng lệnh :

```
$ ifconfig
```

- Để cài đặt địa chỉ IP tĩnh cho hệ thống ta dùng lệnh :

```
$ ifconfig eth0 <địa chỉ IP tĩnh>
```

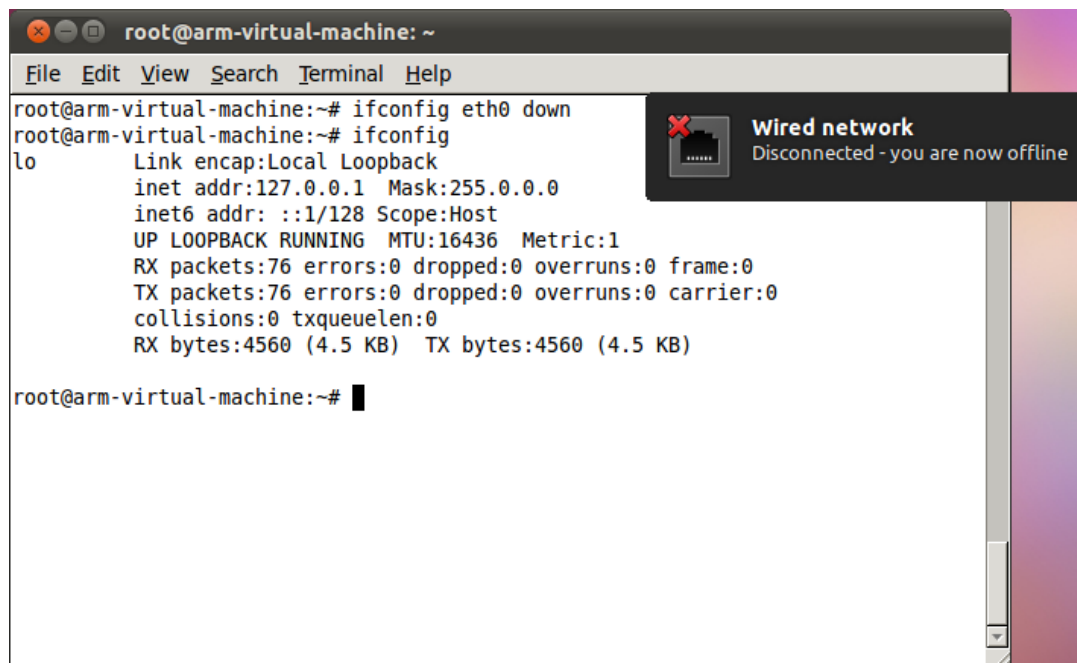
- Để ngừng kích hoạt chế độ chuyển Ethernet ta dùng lệnh :

\$ ifconfig eth0 down

- Khi ta ngừng kích hoạt chế độ truyền Ethernet thì nếu dùng lệnh

\$ ifconfig

Hệ thống sẽ thông báo không có kết nối.



Hình 2.16 : Thông báo cổng Ethernet không có kết nối

- Để kích hoạt chế độ truyền Ethernet ta dùng lệnh :

```
$ ifconfig eth0 up
```

Khi đó hệ thống sẽ thông báo sẵn sàng kết nối

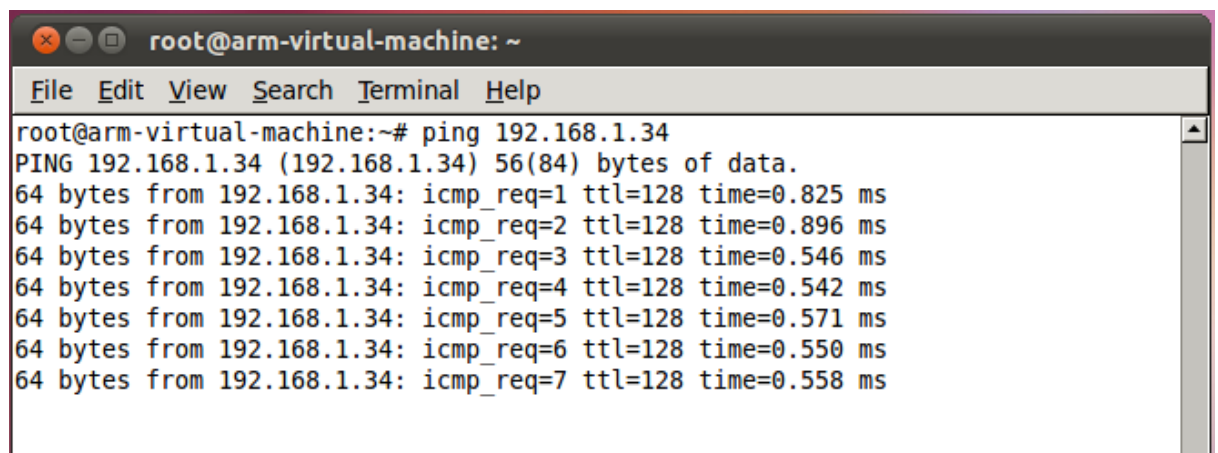


Hình 2.17 : Thông báo cổng Ethernet đã sẵn sàng

- Để kiểm tra hệ thống có kết nối với hệ thống khác hay không ta dùng lệnh :

```
$ ping <địa chỉ IP của hệ thống khác>
```

Nếu thấy có kết nối thì hệ thống sẽ truyền nhận 64 byte dữ liệu liên tục cho máy được kết nối



Hình 2.18 : Hệ thống truyền nhận 64 byte dữ liệu khi thực hiện lệnh ping

Để ngưng quá trình truyền nhận này chúng ta dùng tổ hợp phím Ctrl+C.

13. Tạo một tài khoản người dùng :

- Để xem chúng ta đang đăng nhập với tài khoản người dùng nào, chúng ta dùng lệnh :

\$ whoami

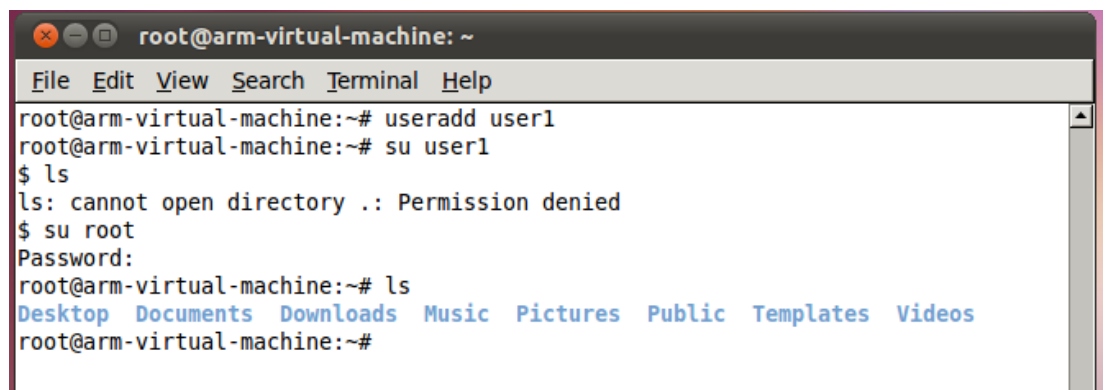
- Chúng ta chỉ tạo được tài khoản người dùng khi chúng ta hoạt động trong tài khoản root. Để tạo ra một tài khoản mới ta dùng lệnh :

\$ useradd <tên tài khoản muốn tạo>

- Để chuyển sang tài khoản người dùng khác ta dùng lệnh :

\$ su <tên tài khoản người dùng>

Ví dụ : ta tạo một tài khoản tên user1 và ta chuyển sang tài khoản user1, rồi ta chuyển sang tài khoản root. Ta làm như sau :



```
root@arm-virtual-machine: ~  
File Edit View Search Terminal Help  
root@arm-virtual-machine:~# useradd user1  
root@arm-virtual-machine:~# su user1  
$ ls  
ls: cannot open directory .: Permission denied  
$ su root  
Password:  
root@arm-virtual-machine:~# ls  
Desktop Documents Downloads Music Pictures Public Templates Videos  
root@arm-virtual-machine:~#
```

Hình 2.19 : Ví dụ về tạo tài khoản người dùng

Ta thấy trong tài khoản user1 ta thực hiện lệnh ls bị lỗi vì các thư mục tập trong thư mục gốc không cho các user khác truy cập. Khi quay trở lại tài khoản root thì chúng ta phải nhập password là root00. Và trong tài khoản root thì ta thực hiện được lệnh ls.

14. Các thao tác với biến môi trường của hệ thống:

- Hệ thống Linux có rất nhiều biến môi trường phục vụ cho quá trình hoạt động của hệ thống. Để xem các biến môi trường của hệ thống ta dùng lệnh sau :

```
$ env
```

- Để in nội dung của một biến môi trường xác định ra màn hình ta dùng lệnh :

```
$ echo $<tên biến môi trường>
```

- Để cài đặt biến môi trường ta dùng lệnh sau :

```
$ export <tên biến môi trường> = <nội dung biến>
```

- Để thêm nội dung vào biến môi trường ta dùng lệnh :

```
$ export <tên biến môi trường> = $<tên biến môi trường> <nội dung muốn thêm vào>
```

Sau khi thêm nội dung vào biến môi trường chúng ta có thể dùng lệnh echo để kiểm tra.

- Gỡ bỏ biến môi trường :

```
$ unset <tên biến môi trường>
```

II. Kết luận:

Hệ điều hành Linux hỗ trợ các thao tác cơ bản và cần thiết trong quá trình người dùng tương tác với hệ điều hành. Trong phần này đã trình bày hầu hết các lệnh căn bản nhất cần thiết cho người học sử dụng Linux.

Như vậy, sau khi đọc xong phần này thì người học đã căn bản sử dụng được hệ điều hành Linux. Nhưng trên đây chỉ là những lệnh căn bản nhất của hệ điều hành Linux, Linux còn có rất nhiều lệnh khác mà người soạn giáo trình không thể liệt kê hết được. Tuy nhiên các lệnh mà được trình bày trong giáo trình này cũng đủ để người học thao tác với Linux trong quá trình học sau này.

C-Trình soạn thảo VI

Nếu chúng ta đã dùng hệ điều hành Window thì chắc hẳn ai cũng quen với trình soạn thảo Notepad. Trong Linux cũng có trình soạn thảo được tích hợp sẵn trong các gói hệ điều hành Linux là trình soạn thảo VI.

Cũng giống như Notepad, trình soạn thảo VI dùng để người dùng soạn dữ liệu, viết chương trình và lưu lại dưới dạng tập tin.

Trong phần này sẽ hướng dẫn cách sử dụng trình soạn thảo VI.

I. Giới thiệu:

VI là chương trình soạn thảo chuẩn trên các hệ điều hành Unix. Nó là chương trình soạn thảo trực quan, hoạt động dưới 2 chế độ: Chế độ lệnh (command line) và chế độ soạn thảo (input mode)

Để soạn thảo tập tin mới hoặc xem, sửa chữa tập tin cũ ta dùng lệnh:

```
/$ vi <tên tập tin>
```

Khi thực hiện, VI sẽ hiện lên màn hình soạn thảo ở chế độ lệnh. Ở chế độ lệnh, chỉ có thể sử dụng các phím để thực hiện các thao tác như: dịch chuyển con trỏ, lưu dữ liệu, mở tập tin...Do đó, bạn không thể soạn thảo văn bản.

Nếu muốn soạn thảo văn bản, bạn phải chuyển từ chế độ lệnh sang chế độ soạn thảo. Chế độ soạn thảo giúp bạn sử dụng bàn phím để soạn thảo nội dung văn bản.

II. Các lệnh thao tác trong VI:

1. Chuyển chế độ trong VI:

- Để chuyển sang chế độ soạn thảo chúng ta có thể dùng một trong hai phím sau trên bàn phím:

i con trỏ sẽ giữ nguyên vị trí.

a con trỏ sẽ dịch sang phải một ký tự.

- Để chuyển ngược lại mode command ta dùng phím ESC
- Để thực hiện các lệnh trong trình soạn thảo VI thì trình soạn thảo VI phải ở chế độ lệnh

2. Nhóm lệnh di chuyển con trỏ :

- h sang trái 1 ký tự
- e đến ký tự cuối cùng của từ gần nhất bên phải
- w đến ký tự đầu tiên của từ gần nhất bên phải
- b đến ký tự đầu tiên của từ gần nhất bên trái
- k lên 1 dòng
- j xuống 1 dòng
-) cuối câu
- (đầu câu
- } đầu đoạn văn
- { cuối đoạn văn

3. Nhóm lệnh xóa:

- dw xóa 1 từ
- d^ xóa ký tự từ con trỏ đến đầu dòng
- d\$ xóa ký tự từ con trỏ đến cuối dòng
- 3dw xóa 3 từ
- dd xóa dòng hiện hành
- 5dd xóa 5 dòng
- x xóa 1 ký tự

Các từ được cách nhau bởi khoảng trắng.

4. Nhóm lệnh thay thế:

- cw thay thế 1 từ
- 3cw thay thế 3 từ
- cc dòng hiện hành
- 5cc 5 dòng

5. Nhóm lệnh copy, past, undo:

Để copy ta dùng lệnh y và để paste ta dùng lệnh p

- y\$ copy từ vị trí hiện tại của cursor đến cuối cùng
- yy copy toàn bộ dòng tại vị trí cursor
- 3yy copy 3 dòng liên tiếp
- u Undo lại thao tác trước đó

6. Thao tác trên tập tin :

:x lưu và thoát khỏi chế độ soạn thảo
:wq lưu và thoát khỏi chế độ soạn thảo
:w lưu vào tập tin mới
:q thoát nếu ko có thay đổi
:q! thoát không lưu

- Sau khi gõ các lệnh trên thì nhấn Enter để thực thi lệnh.

III. Kết luận:

VI là trình soạn thảo của Linux, VI có hai chế độ làm việc là chế độ lệnh và chế độ soạn thảo. Các thao tác trên VI rất khác so với các tác của các trình soạn thảo quen thuộc nên làm cho người dùng cảm thấy khó khăn trong việc soạn thảo dữ liệu với VI.

Vì tính phức tạp của trình soạn thảo VI, nên trong thực tế trình soạn thảo VI chỉ dùng vào mục đích soạn thảo tập tin có nội dung ngắn hoặc là để sửa nội dung tập tin trong trường hợp cần sửa một cách nhanh chóng trong hệ điều hành Linux. Còn đối với việc soạn các tập tin có nội dung phức tạp, dài thì các lập trình viên vẫn thường dùng các trình soạn thảo quen thuộc của Window như Notepad, Wordpad, Microsoft Word,....

Như vậy, trong bài này chúng ta đã làm quen với hệ điều hành Linux, một trong những hệ điều hành được sử dụng phổ biến hiện nay. Hiểu biết về Linux sẽ tạo nền tảng cho chúng ta tìm hiểu sâu về hệ thống nhúng.

Trong bài sau chúng ta sẽ tìm hiểu về hệ thống nhúng là kit nhúng KM9260.

BÀI 3

**HỆ THỐNG PHẦN CỨNG VÀ PHẦN MỀM
TRONG KIT NHÚNG KM9260**

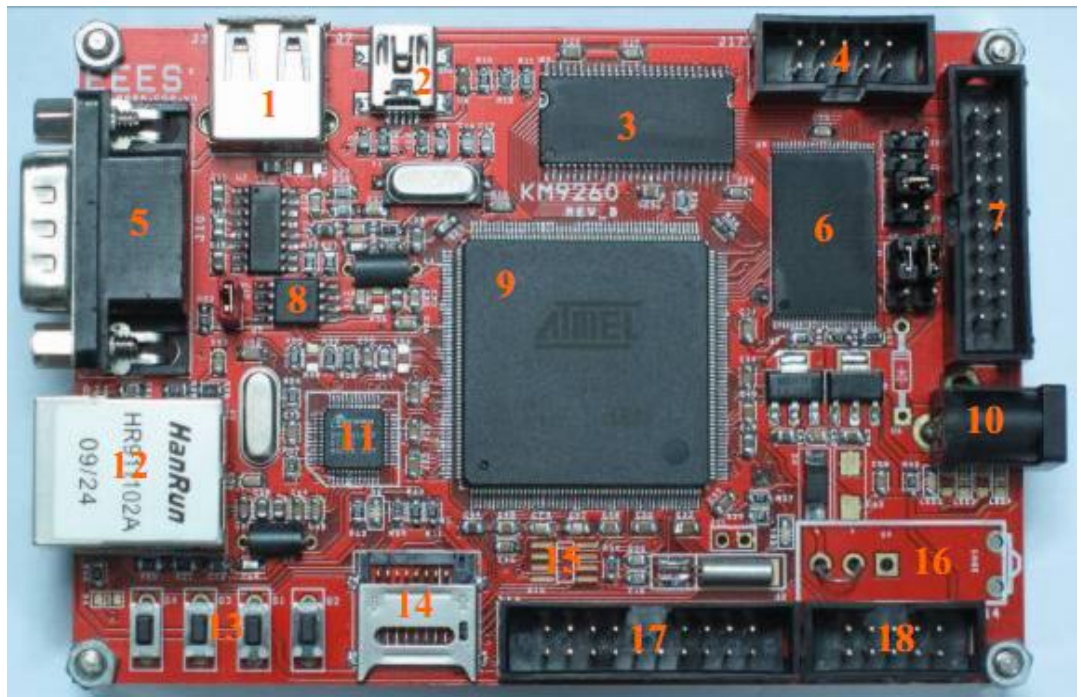
A-Phần cứng hệ thống nhúng trong kit KM9260:

Một hệ thống nhúng bao gồm hai phần: phần cứng và phần mềm. Trong đó phần cứng là nền tảng, là điều kiện cần để một hệ thống nhúng có thể tồn tại. Để có thể sử dụng được một hệ thống nhúng thì chúng ta phải hiểu được phần cứng của nó

Trong hệ thống nhúng ngoài vi xử lý là nòng cốt thì bên cạnh đó còn có nhiều linh kiện khác: chip nhớ, các thiết bị ngoại vi, ... Trong phần này chúng ta sẽ tìm hiểu phần cứng của Kit KM9260 gồm những linh kiện nào, chức năng của các linh kiện, vị trí của từng linh kiện trên board.

Vì phần cứng hệ thống nhúng đa số là do các công ty sản xuất thiết kế, thi công nên với tư cách là một người nghiên cứu để sử dụng thì trong giáo trình này chúng ta không đi sâu vào mạch nguyên lý của kit nhúng.

Hình dạng của board Kit KM9260



Hình 2.20 : Hình dạng kit KM9260

Các thành phần trên board bao gồm:

STT	KÝ HIỆU	TÊN
1	J3	Cổng USB (loại A)
2	J7	Cổng USB Devices (loại B)
3	U2	MT48LC16M16A2, SDRAM 256Mb (32MB) 133Mhz
4	J17	Kết nối mở rộng SCI
5	J10	Cổng truyền dữ liệu nối tiếp BD9
6	U8	K9F2G08UOM, NAND Flash (256MB)
7	J5	Giao tiếp JTAG ICE
8	U9	Serial dataflash (512kB)
9	U1	AT91SAM9260, 16/32 bit ARM926EJ-S 180Mhz
10	J12	Jack nguồn 5VDC
11	U5	DM9161EA, Ethernet 10/100 Full-Duplex
12	RJ1	Cổng Giao tiếp Ethernet (RJ45)
13	S1, S2, S3, S4	Các nút nhấn
14	U10	Khe thẻ nhớ MicroSD
15	U11	I2C EEPROM
16	J12	Công tắc nguồn
17	J16	Kết nối mở rộng Uart, Adc, Twi
18	J14	Kết nối mở rộng SPI

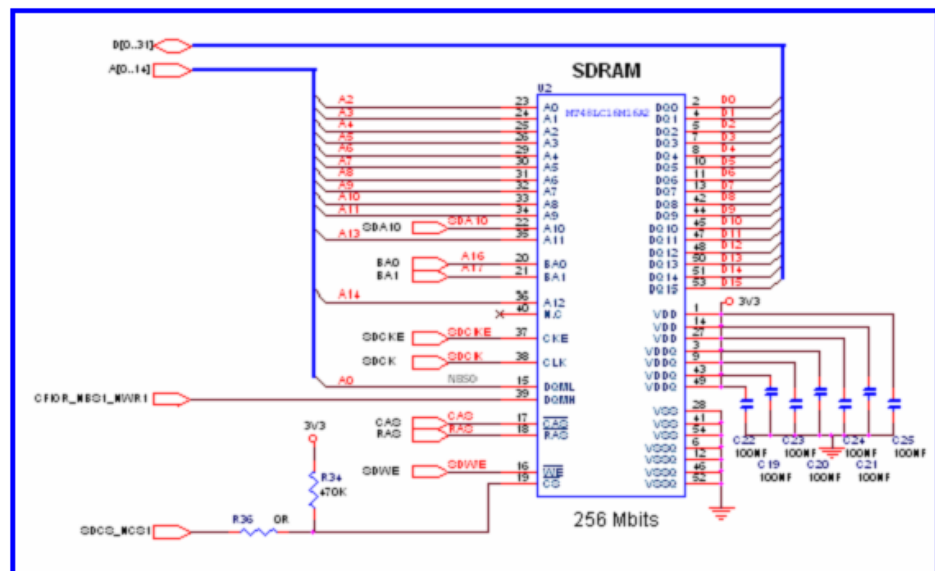
Bảng 2.5: Các linh kiện trên kit KM9260

I. Cpu (C1):

Sử dụng vi điều khiển AT91SAM9260 đã được giới thiệu trong phần trước.

II. Bộ nhớ:

1. Sdram (U2):



Hình 2.21 : Sơ đồ kết nối SDRAM

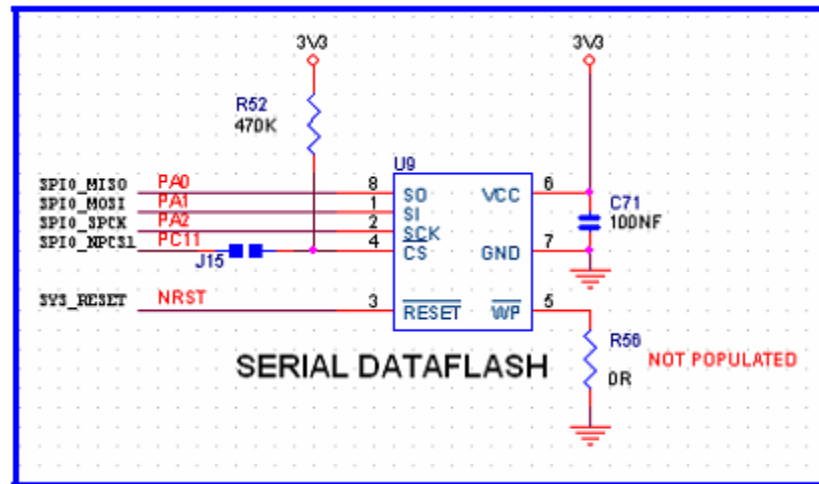
- Bộ nhớ chính sử dụng SDRAM bus 133Mhz, SDRAMC được cấu hình với bus data 16 bit.
- Bảng sau trình bày thông số bảng đồ vùng nhớ của SDRAM trong hệ thống.

Mã số linh kiện	MT48LC16M16A2 TC75
Chân chọn chip	NCS1
Địa chỉ đoạn	0x20000000
Dung lượng	0x2000000 (32MB)

Bảng 2.6: Thông số của SDRAM

- Mã số linh kiện là số được in trên thân của chip, mã số này do nhà sản xuất quy định.
- Chân chọn chip là NCS1 có nghĩa là chân CS của chip được nối với chân SD_CS_NCS1 của vi xử lý.
- Địa chỉ đoạn là địa chỉ mà vi xử lý gán cho chip tương ứng với NCS1.
- Dung lượng của SDRAM là 32 MB

2. Serial dataflash (U9):

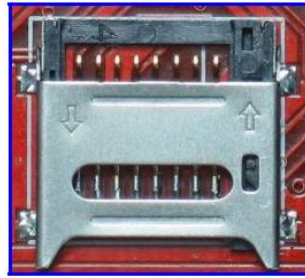


Hình 2.22 : Sơ đồ kết nối Serial DataFlash

- Board sử dụng chip nhớ serial dataflash kết nối qua đường SPI0 (slot CS1).
- AT91Bootstrap, các biến môi trường (U-Boot's Environment Variables), U-Boot được lưu trữ trong serial dataflash. Các phân vùng chứa các bootloader được thể hiện bởi bảng sau:

Mã số linh kiện	AT45DB041D-SU	
Chân chọn chip	NCS2	
Địa chỉ đoạn	0xD0000000	
Dung lượng	0x80000 (512kB)	
Offset	Phân Vùng	Phần mềm lưu
0x00000000	0	Bootstrap
0x00004200	1	Environment
0x00008400	2	U-Boot

Bảng 2.7: Thông số của Serial DataFlash

4. Thẻ nhớ MicroSD (U10):

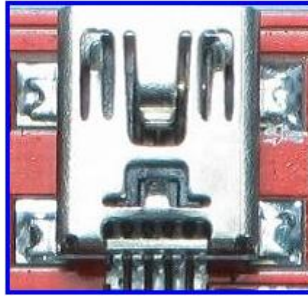
Hình 2.24 : Khe cắm thẻ nhớ Micro SD

- Do có hỗ trợ microSD, có thể thay thế vai trò của NAND Flash cho việc lưu trữ kernel Linux và rootfs. Phân vùng đầu tiên (first partition) được format theo định dạng FAT, phân vùng thứ 2 được định dạng theo ext2 hoặc ext3 dùng để chứa rootfs. Để load kernel Linux từ MicroSD vào SDRAM đòi hỏi U-Boot phải hỗ trợ mmc sub system command set. MicroSD thường được dùng để boot Linux có rootfs dung lượng lớn, ví dụ như Debian distribution.

III. Kết nối ngoại vi:**1. Cổng USB (J13):**

Hình 2.25 : Cổng USB

Cổng USB tốc độ truy xuất 2.0, tương tự MicroSD, hệ thống có thể boot Linux thông qua ổ đĩa di động USB..

2. Cổng thiết bị USB (J7):

Hình 2.26 : Cổng USB Device

Với USB device connector, ta có thể biến board nhúng thành các thiết bị USB. MCU AT91SAM9260 cho phép ta truy xuất đến tất cả các vùng nhớ trong hệ thống thông qua chương trình ứng dụng trên máy tính SAMBA, khi đó dây cable USB này được dùng đến.

3. Cổng truyền dữ liệu nối tiếp DB9 (J10):

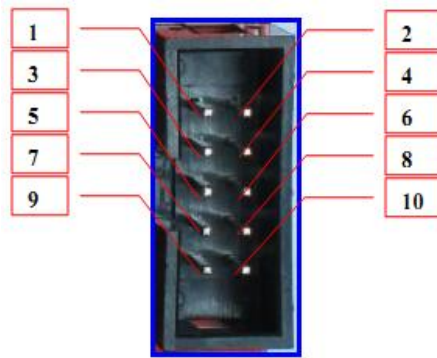
Hình 2.27 : Cổng truyền dữ liệu nối tiếp DB9

AT91SAM9260 có tích hợp cổng RS232. KM9260 dùng cổng RS232 này cho việc hiển thị, xuất nhập với console chính của Linux.

4. Cổng kết nối Ethernet (RJ1):

Hình 2.28 : Cổng kết nối Ethernet

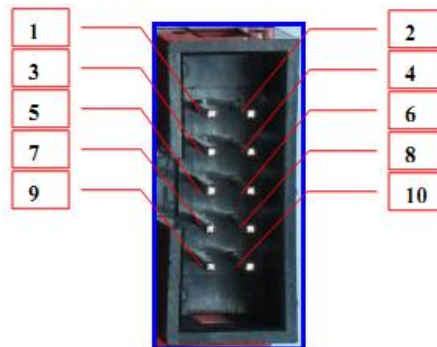
AT91SAM9260 có thể kết hợp với chip Fast Ethernet PHY DM9161AEP mang lại cho hệ thống tính năng mạnh mẽ về các ứng dụng mạng. KM9260 có thể sử dụng như hệ thống webserver nhúng, sử dụng trong hệ thống thu thập đo lường, điều khiển từ xa...

5. Kết nối mở rộng SCI (J17):*Hình 2.29 : Khe cắm kết nối mở rộng SCI*

SỐ CHÂN	TÍN HIỆU	CHÂN VXL	SỐ CHÂN	TÍN HIỆU	CHÂN VXL
1	GND	-	2	5V	-
3	GND	-	4	3V3	-
5	TK0	23 (PB16)	6	TF0	26 (PB17)
7	TD0	27 (PB18)	8	RD0	28 (PB19)
9	RK0	163 (PB20)	10	RF0	164 (PB21)

Bảng 2.9: Sơ đồ chân của khe cắm kết nối mở rộng SCI

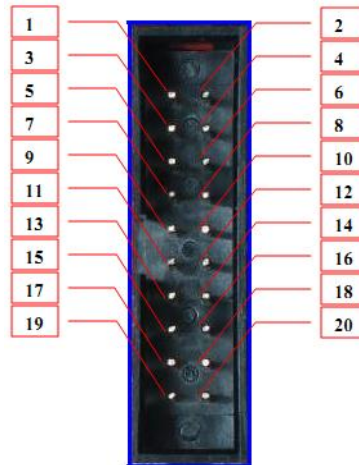
Connector cho phép ta có thể mở rộng kết nối với các thiết bị I2S audio codec, truyền nhận dữ liệu 32bit stream (High-speed Continuous Data Stream)...

6. Kết nối mở rộng SPI (J14):*Hình 2.30 : Khe cắm kết nối mở rộng SPI*

SỐ CHÂN	TÍN HIỆU	CHÂN VXL	SỐ CHÂN	TÍN HIỆU	CHÂN VXL
1	GND	-	2	3V3	-
3	SPI1_MISO	9 (PB0)	4	SPI1_MOSI	10 (PB1)
5	SPI1_SPCK	11 (PB2)	6	SPI1_NPCS0	12 (PB3)
7	SPI1_NPCS1	67 (PC5)	8	SPI1_NPCS2	62 (PC4)
9	GPIO	63 (PC6)	10	GPIO	64 (PC7)

Bảng 2.10: Sơ đồ chân của khe cắm kết nối mở rộng SPI

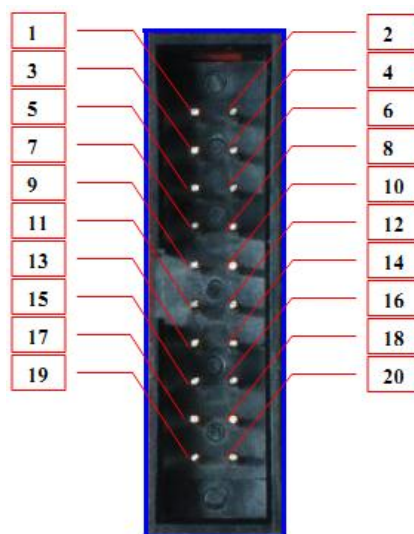
Connector mở rộng cho SPI, bao gồm 3 đường chip select tương ứng cho 3 slot CS0, CS1, CS2.

7. Kết nối mở rộng UART, ADC, TWI (J16):*Hình 2.31: Khe cắm kết nối mở rộng Uart, Adc, Twi*

SỐ CHÂN	TÍN HIỆU	CHÂN VXL	SỐ CHÂN	TÍN HIỆU	CHÂN VXL
1	5V	-	2	3V3	-
3	AVDD	-	4	GND	-
5	AGND	-	6	VREFP	-
7	AD0	158 (PC0)	8	AD1	159 (PC1)
9	IRQ1	127 (PC15)	10	UART_TXD0	15 (PB4)
11	UART_RXD0	16 (PB5)	12	UART_TXD1	17 (PB6)
13	UART_RXD1	18 (PB7)	14	UART_TXD2	19 (PB8)
15	UART_RXD2	20 (PB9)	16	UART_TXD3	161 (PB10)
17	UART_RXD3	162 (PB11)	18	TWD	208 (PA23)
19	TWCK	1 (PA24)	20	GPIO	23 (PB16)

Bảng 2.11: Sơ đồ chân của khe cắm kết nối mở rộng Uart, Adc, Twi

Connector mở rộng cho các cổng giao tiếp serial bao gồm UART, TWI, bộ chuyển đổi ADC, GPIO, ngắt ngoài IRQ1.

8. Giao tiếp JTAG ICE (J5) :*Hình 2.32: Khe cắm giao tiếp JTAG ICE*

SỐ CHÂN	TÍN HIỆU	CHÂN VXL	SỐ CHÂN	TÍN HIỆU	CHÂN VXL
1	3V3	-	2	3V3	-
3	NTRST	35	4	GND	-
5	TDI	30	6	GND	-
7	TMS	31	8	GND	-
9	TCK	34	10	GND	-
11	RTCK	37	12	GND	-
13	TDO	29	14	GND	-
15	NRST	36	16	GND	-
17	NC	-	18	GND	-
19	NC	-	20	GND	-

Bảng 2.12: Sơ đồ chân của khe cắm giao tiếp JTAG ICE

Cổng JTAG ICE theo chuẩn 20 pin cho phép nạp chương trình và debug hệ thống.

IV. Nút nhấn:

1. Nút Reset (S2):



Hình 2.33: Nút reset

Reset hệ thống, tích cực mức thấp.

2. Nút Wake up (S1):



Hình 2.34: Nút Wake up

Nút có tác dụng đánh thức hệ thống từ trạng thái power down.

3. Nút ứng dụng 1 (S3):



Hình 2.35: Nút ứng dụng 1

User button, kết nối với PC15 (chân 127) của AT91SAM9260 MCU. Tích cực mức thấp.

4. Nút ứng dụng 2 (S4):



Hình 2.36: Nút ứng dụng 2

User button, kết nối với PC8 (pin 61) của AT91SAM9260 MCU. Tích cực mức thấp.

V. Led hiển thị:**1. Led hiển thị trạng thái của hệ thống:**

TÊN LED	CHỨC NĂNG	TRẠNG THÁI	Ý NGHĨA
LED1	Shutdown	On	VXL Trạng thái Power down
		Off	VXL không hoạt động
LED2	Power	On	Bật nguồn
		Off	Tắt nguồn
D6	Duplex	On	Ethernet PHY full-duplex
		Off	Ethernet PHY half-duplex

*Bảng 2.13: Ý nghĩa của các Led hiển thị trạng thái của hệ thống***2. Led ứng dụng:**

Ký hiệu là D4, được nối với chân PA6 (chân 185). Tích cực mức thấp.

VI. Jumper :**1. Jumper chọn chip :**

KÝ HIỆU	CHỨC NĂNG
J15	Chọn chip Serial dataflash
J13	Chọn chip NAND Flash

Bảng 2.14: Chức năng của J13 và J15*Hình 2.37: Serial DataFlash và Jumper chọn Serial DataFlash*

Jumper chọn chip được thiết kế hỗ trợ cho việc khôi phục boot loader sau khi người dùng ghi chương trình không phù hợp vào vùng Bootstrap hoặc U-Boot của hệ thống

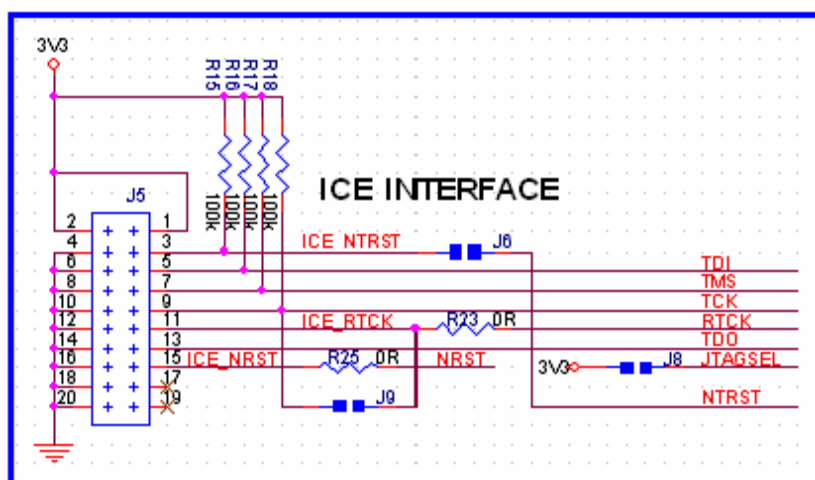
2. Jumper chọn cấu hình hệ thống:

KÝ HIỆU	CHỨC NĂNG	VỊ TRÍ	Ý NGHĨA	MẶC ĐỊNH
J1	Chọn tần số hoạt động	1-2	Dùng dao động nội RC	2-3
		2-3	Dùng thạch anh 32,768Khz	
J2	Chọn nguồn cho vi xử lý	1-2	Dùng nguồn pin	1-2
		2-3	Dùng nguồn chính 1,8 V	

Bảng 2.15 : Chức năng của J1 và J2

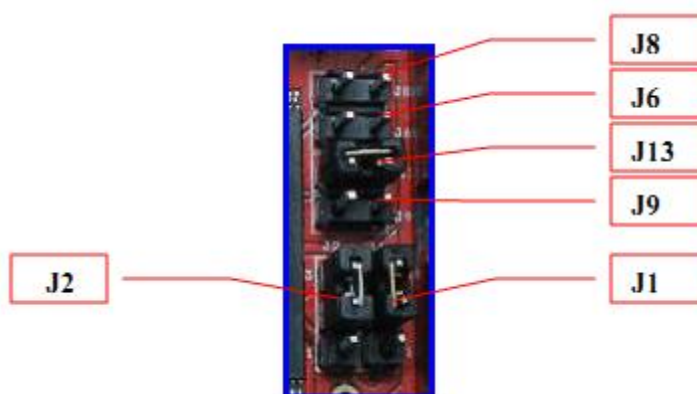
3. Các Jumper liên quan đến JTAG

- Sơ đồ mạch FTAG INTERFACE bao gồm các jumper liên quan đến JTAG như sau :



Hình 2.38: Sơ đồ kết nối của giao tiếp JTAG ICE

- Vị trí của các Jumper J1, J2, J6, J8, J9, J13 được thể hiện ở hình sau :



Hình 2.39: Các jumper J1, J2, J6, J8, J9, J13

VII. Tổng kết:

Phần cứng hệ thống nhúng gồm nhiều thành phần : vi xử lý, chip nhớ, ngoại vi, nút nhấn, led báo, jump được kết nối với nhau tạo thành một thể thống nhất. Mỗi thành phần đều có một chức năng riêng.

Để thuận lợi cho việc tìm hiểu sâu hơn về kit KM9260 thì người học sau khi đọc phần này nên cố gắng nhận diện được các linh kiện trên kit, xác định được vị trí, chức năng của từng linh kiện.

Như đã nói ở phần trên, chúng ta sẽ không tìm hiểu sâu về phần cứng của kit KM9260. Thay vào đó chúng ta sẽ tìm hiểu kỹ hơn về các phần mềm hệ thống nhúng trong phần tiếp theo của giáo trình.

B-Phần mềm hệ thống nhúng trong kit KM9260:

Một trong những thế mạnh của hệ thống nhúng chính là phần mềm. Chính phần mềm đã giúp hệ thống nhúng tới gần hơn người sử dụng. Ngoài ra phần mềm của hệ thống nhúng giúp cho hệ thống hoạt động ổn định, quản lý được các sự cố, giúp cho hệ thống hoạt động hết hiệu năng của mình.

Ngày nay cùng với sự phát triển về phần cứng thì phần mềm của hệ thống nhúng cũng phát triển mạnh mẽ. Nó giúp cho hệ thống nhúng giải quyết được nhiều ứng dụng trong thực tế, giúp hệ thống nhúng ngày càng trở nên thông minh.

Có thể nói nếu phần cứng là phần xác của hệ thống nhúng thì phần mềm chính là phần hồn của hệ thống nhúng.

Trong phần này chúng ta sẽ tìm hiểu các phần mềm cơ bản cần phải có của một hệ thống nhúng, trình tự thực hiện các phần mềm này, hay nói cách khác là trình tự thực thi các phần mềm hệ thống của hệ thống nhúng.

I. Các phần mềm trong hệ thống nhúng:

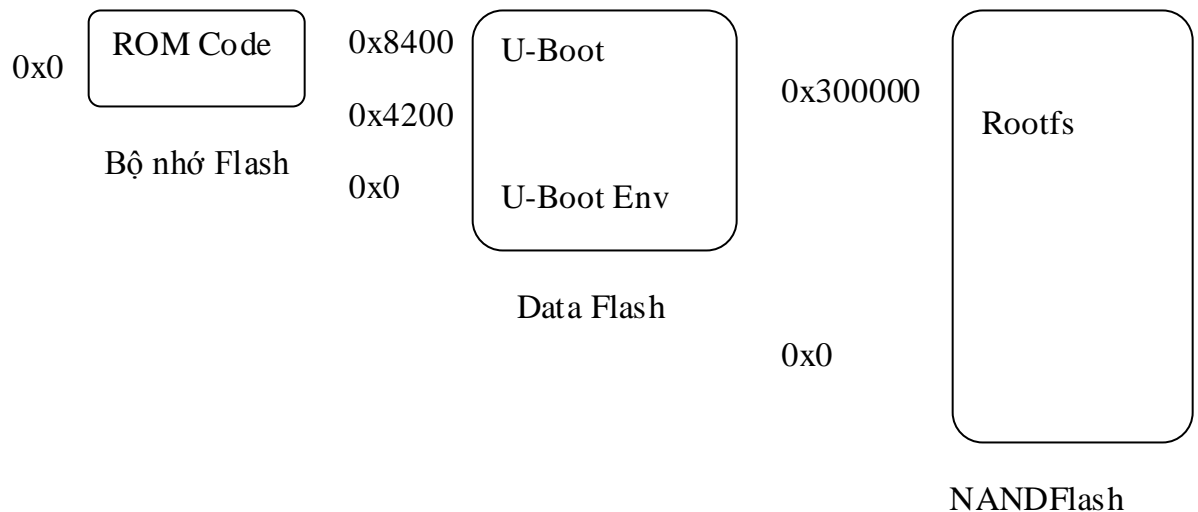
Một hệ thống nhúng thông thường có các phần mềm sau:

- *ROM Code*: là phần mềm do nhà sản xuất viết và nạp sẵn trong bộ nhớ Flash của Vi Xử Lý. ROM Code có chức năng khởi động cho hệ thống.
- *Bootstrapcode*: có chức năng cấu hình phần cứng (các thanh ghi điều khiển UART_DB, SDRAM, NANDFlash, SPIx giao tiếp DataFlash) sau đó chép U-Boot từ Flash lên SDRAM và chuyển quyền thực thi cho U-Boot.
- *U-Boot*: khi thực thi sẽ cấu hình phần cứng (USB, Network, SD/MMC, PIO, USB, ...) có rất nhiều hàm hỗ trợ ở U-boot. Tiếp theo thực thi các lệnh trong U-Boot từ màn hình Terminal hoặc từ biến môi trường đã được người dùng cài đặt cấu hình từ trước. Các lệnh mà người dùng cấu hình sẵn thường là lệnh chép Kernel từ Flash, Network, SD Card, ... lên SDRAM sau có khởi động Kernel.
- *Kernel*: là hạt nhân của hệ điều hành, khi chạy sẽ khởi động các drivers cần thiết để giao tiếp với phần cứng, sau đó kết gán với Rootfs.
- *Rootfs (root file system)*: chứa các ứng dụng và thực thi chúng, sử dụng các dịch vụ của kernel để triệu gọi các hàm ở lớp driver khi thực thi.

II. Phân vùng trên Kit KM9260:

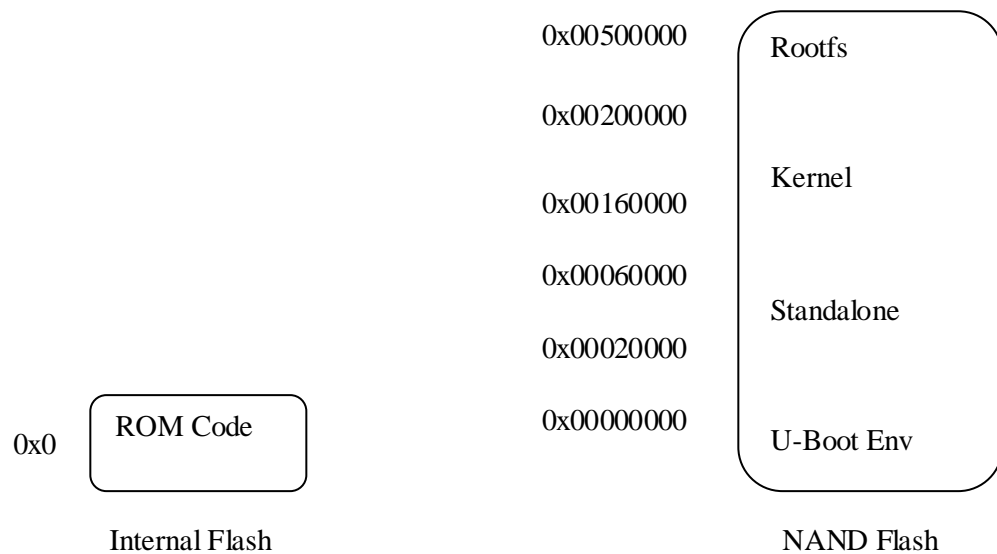
Phân vùng trên Kit KM9260 là vị trí lưu trữ các phần mềm trong hệ thống nhúng.

Có 2 loại phân vùng của KIT KM9260 tùy theo quy định của nhà sản xuất:

1. Phân vùng loại 1 trên Kit KM9260:

Hình 2.40: Phân vùng loại 1 trên kit KM9260

- ROM Code được nhà sản xuất nạp sẵn trong bộ nhớ nội trong vi xử lý. Địa chỉ bắt đầu nạp là 0x0 nghĩa là ô nhớ đầu tiên.
- AT91 Bootstrap được lưu trữ trên Data Flash. Địa chỉ bắt đầu là 0x0.
- U-Boot Env là các biến môi trường của U-Boot để cấu hình cho quá trình khởi động cho hệ thống. Các biến này được lưu trữ trên Data Flash. Địa chỉ bắt đầu là 0x4200.
- U-Boot cũng được lưu trữ trên Data Flash với địa chỉ bắt đầu là 0x8400.
- Kernel được nạp trên NAND Flash với địa chỉ bắt đầu là 0x0.
- Rootfs cũng được nạp trên NAND Flash với địa chỉ bắt đầu là 0x300000.

2. Phân vùng loại 2 của KIT KM9260 :

Hình 2.41: Phân vùng loại 2 trên kit KM9260

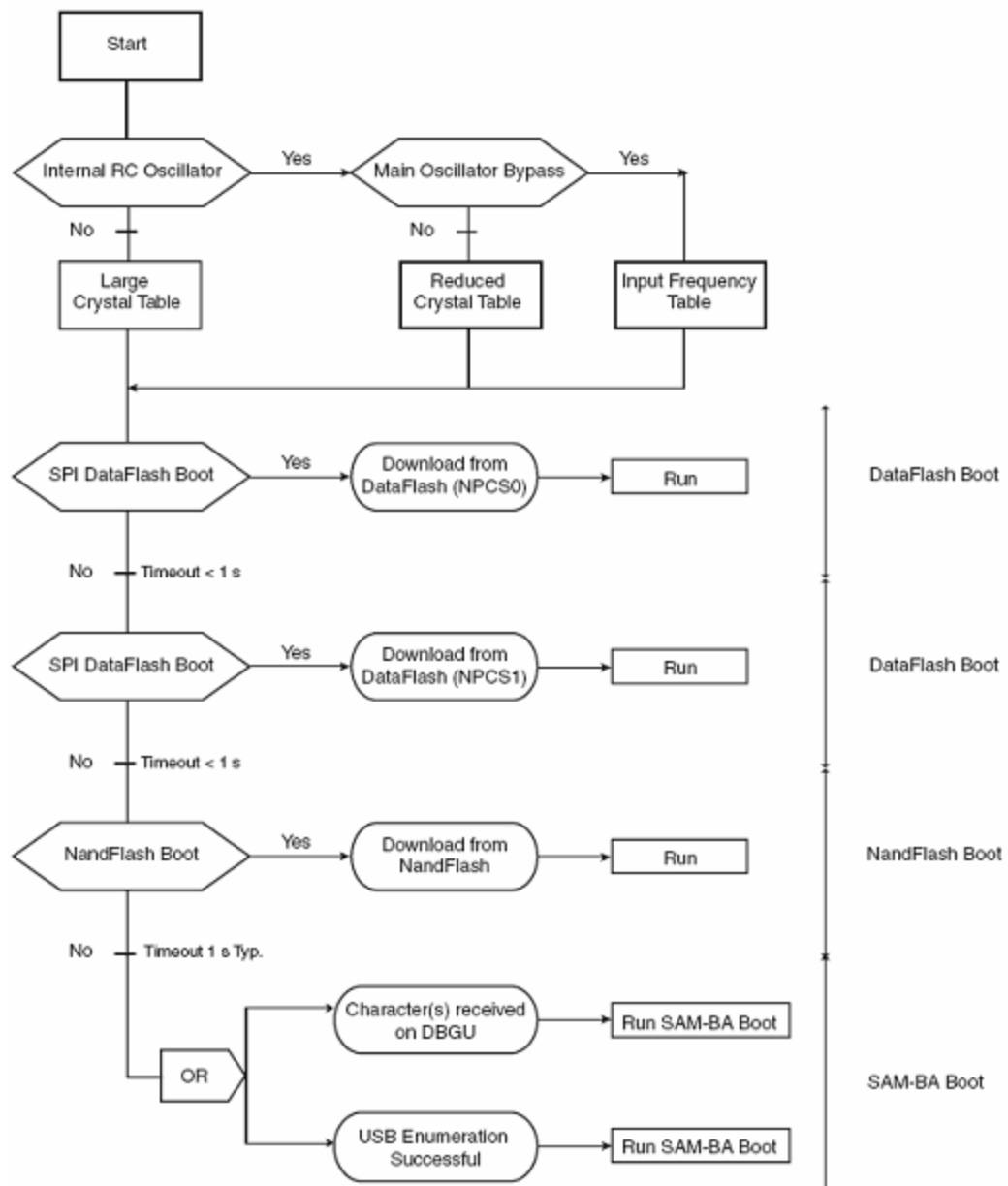
- ROM Code được nhà sản xuất nạp sẵn trong bộ nhớ nội trong Vi xử Lý. Địa chỉ bắt đầu nạp là 0x0 nghĩa là ô nhớ đầu tiên.
- AT91 Bootstrap được lưu trữ trên NAND Flash. Địa chỉ bắt đầu là 0x00000000.
- U-Boot cũng được lưu trữ trên NAND Flash với địa chỉ bắt đầu là 0x00020000.
- U-Boot Env là các biến môi trường của U-Boot để cấu hình cho quá trình khởi động cho hệ thống. Các biến này được lưu trữ trên NAND Flash. Địa chỉ bắt đầu là 0x00060000.
- Standalone : là vùng nhớ lưu các chương trình chạy đơn nhiệm (là các chương trình ứng dụng được lập trình cho vi xử lý chạy là không cần dùng hệ điều hành. Vùng nhớ này ở NAND Flash với địa chỉ bắt đầu là 0x00160000.
- Kernel cũng được nạp trên NAND Flash với địa chỉ bắt đầu là 0x00200000.
- Rootfs cũng được nạp trên NAND Flash với địa chỉ bắt đầu là 0x00500000.

Lưu ý : các địa chỉ nói trên đây là các địa chỉ offset

III. Boot loader cho Kit KM9260:

Boot loader có vai trò quan trọng trong hệ thống nhúng. Boot loader là trình tự thực thi các phần mềm nói trên của vi xử lý trong quá trình khởi động của hệ thống.

Tùy theo loại vi xử lý khác nhau mà ta có cơ chế, trình tự boot khác nhau. Đối với AT91SAM9260, khi cấp nguồn AT9260 chạy chương trình boot bên trong ROM (được xây dựng sẵn trong quá trình sản xuất chip). Hình sau thể hiện lưu đồ ROM boot của AT91SAM9260:



Hình 2.42: Lưu đồ ROM Boot của AT91SAM9269

B1> Chương trình chạy với bộ dao động nội RC, hoặc với bộ dao động thạch anh 32,768 Khz (được chọn bởi Jumper J1).

Nếu chương trình chạy với bộ dao động nội RC và bỏ qua bộ dao động chính thì chương trình boot chỉ hỗ trợ được các tần số ngõ vào XIN sau: 1Mhz; 2 Mhz; 6 Mhz; 12 Mhz; 25 Mhz; 50 Mhz.

Nếu chương trình chạy với bộ dao động nội RC và bộ dao động chính được kích hoạt thì chương trình boot có thể hỗ trợ được các thạch anh ở ngõ vào XIN, XOUT tạo tần số sau: 3 Mhz; 6 Mhz; 18,432 Mhz.

Nếu chương trình chạy với bộ dao động thạch anh 32,768 Khz thì chương trình boot sẽ hỗ trợ các thạch anh ở ngõ vào XIN, XOUT có tần số sau:

3 Mhz	3,2768 Mhz	3,6864 Mhz	3,84 Mhz	4 Mhz
4,433619 Mhz	4,9152 Mhz	5 Mhz	5,24288 Mhz	6 Mhz
6,144 Mhz	6,4 Mhz	6,5536 Mhz	7,159090 Mhz	7,3728 Mhz
7,864320 Mhz	8 Mhz	9,8304 Mhz	10 Mhz	11,05920 Mhz
12 Mhz	12,288 Mhz	13,56 Mhz	14,31818 Mhz	14,7456 Mhz
16 Mhz	16,367667 Mhz	17,734470 Mhz	18,432 Mhz	20 Mhz

Bảng 2.16: Các tần số của thạch anh dao động chính mà chương trình boot có thể hỗ trợ khi dùng dao động chậm là thạch anh 32,768 KHz

Như vậy, nếu chúng ta để Jump J1 ở vị trí 1-2 thì chương trình sẽ chạy với bộ dao động nội RC và bộ dao động chính được kích hoạt với thạch anh ngõ vào XIN là 18,432 Mhz. Nếu chúng ta để J1 ở vị trí 2-3 thì chương trình sẽ chạy với bộ dao động thạch anh 32,768 Khz và bộ dao động chính với thạch anh ngõ vào là 18,432 Mhz.

B2> Kiểm tra sự tồn tại của chương trình AT91BootStrap trong SPI serial dataflash device (NPCS0) hay không, nếu tồn tại, MPU thực hiện chép mã thực thi của AT91BootStrap từ SPI serial dataflash device vào SRAM nội của MPU, sau đó thực thi lệnh nhảy đến địa chỉ đầu tiên của SRAM để thực thi chương trình AT91BootStrap. Nếu không tồn tại chương trình bootstrap trong SPI serial dataflash device, MPU thực hiện bước B3 sau đây.

B3> Kiểm tra sự tồn tại của chương trình AT91BootStrap trong SPI serial dataflash device (NPCS1/NCS2) hay không, nếu tồn tại, MPU thực hiện chép mã thực thi của AT91BootStrap từ SPI serial dataflash device vào SRAM nội của MPU, sau đó thực thi lệnh nhảy đến địa chỉ đầu tiên của SRAM để thực thi chương trình AT91BootStrap.

Nếu không tồn tại chương trình bootstrap trong SPI serial dataflash device, MPU thực hiện bước B4 sau đây.

B4> Kiểm tra sự tồn tại của chương trình AT91BootStrap trong NAND FLASH device hay không, nếu tồn tại, MPU thực hiện chép mã thực thi của AT91BootStrap từ NAND FLASH vào SRAM nội của MPU, sau đó thực thi lệnh nhảy đến địa chỉ đầu tiên của SRAM để thực thi chương trình AT91BootStrap. Nếu không tồn tại chương trình bootstrap trong SPI serial dataflash device, MPU thực hiện bước B5 sau đây.

B5> Nếu MPU nhận được ký tự bất kỳ từ bàn phím máy tính (qua cổng DBGU). Hoặc khi cắm cable USB vào máy tính chương trình sẽ nhảy sang SAM-BA boot.

- Nếu chương trình AT91BootStrap được thực thi thì AT91BootStrap sẽ tiến hành Boot Kernel Linux. Quá trình này được thực hiện qua 3 bước như sau:

B1> KM9260 thực hiện khởi động PLL cho system clock, khởi động SDRAM controller, load u-boot.bin từ AT45DB041D-SU tại offset 0x8400 vào địa chỉ 0x21F00000 của SDRAM. Sau đó thực hiện lệnh nhảy vào vùng SDRAM để chạy chương trình U-Boot.

B2> U-Boot load kernel uImage từ nhiều nguồn khác nhau (TFTP, NAND FLASH...) chép vào địa chỉ 0x20000000 trên SDRAM, sau đó thực hiện lệnh boot hệ điều hành Linux.

B3> Cuối cùng hệ điều hành sẽ kết gán với Rootfs.

IV. Tổng kết:

Phần mềm hệ thống nhúng gồm có 5 thành phần như sau: romcode, bootstrapcode, u-boot, kernel, rootfs. Mỗi thành phần có một nhiệm vụ riêng. Nếu thiếu một trong các phần mềm trên thì hệ thống nhúng không thể nào hoạt động được.

Các phần mềm nói trên là các phần mềm hệ thống, do nhà sản xuất viết ra đi kèm theo kit. Các phần mềm này sẽ giúp người dùng thực thi các phần mềm ứng dụng sau này. Cách thức viết các chương trình ứng dụng sẽ được trình bày một cách kỹ càng trong các phần sau của giáo trình.

Khi kit mới được sản xuất thì các phần mềm hệ thống trên nằm bên ngoài kit. Vậy làm thế nào mà nạp được chúng vào kit? Đó là nhờ sự hỗ trợ của các phần mềm lập trình nhúng. Trong phần sau chúng ta sẽ tìm hiểu về các phần mềm hỗ trợ trong quá trình sử dụng kit.

BÀI 4

PHẦN MỀM HỖ TRỢ LẬP TRÌNH NHÚNG

Khi một hệ thống nhúng ra đời thì nhà sản xuất luôn đính kèm theo các phần mềm hỗ trợ cho quá trình sử dụng hệ thống. Các phần mềm này giúp người dùng sử dụng hệ thống nhúng một cách dễ dàng và linh hoạt.

Như phần trước đã trình bày, để nạp các phần mềm hệ thống vào vùng nhớ tương ứng trên kit thì chúng ta có một phần mềm hỗ trợ đó là phần mềm SAMBA. Ngoài ra chúng ta còn có phần mềm putty.exe có vai trò tạo ra màn hình terminal để thao tác với kit. Phần mềm tftpd32.exe giúp chúng ta chép một tập tin vào vùng nhớ ứng dụng của kit. Ngoài ra còn có phần mềm SSH Secure Shell Client để chép dữ liệu qua lại giữa Window và Linux. Phần mềm tạo máy tính ảo Vmware Workstation.

Tất cả các phần mềm này sẽ được trình bày một cách tỉ mỉ trong phần này. Mỗi phần mềm chúng ta sẽ tìm hiểu cách cài đặt, chức năng và cách sử dụng của chúng.

A- Chương trình máy tính ảo VMware Workstation:

I. Giới thiệu:

VMware Workstation là một phần mềm tạo máy ảo chuyên nghiệp, hỗ trợ nhiều chức năng hơn hẳn các phần mềm trong ngành như: Virtual PC 2007, VirtualBox, ... Máy ảo thực chất là một phần mềm chạy trên hệ điều hành hiện thời trên máy thật (gọi là Hệ điều hành chủ), nó sẽ tạo một chiếc máy tính “thực” hoàn toàn để ta có thể cài một hệ điều hành khác lên (gọi là hệ điều hành khách – Guest OS) và chạy như thể chạy từ chính máy tính của mình. Như thế ta có thể chạy Linux trong Windows một cách dễ dàng. Ta có thể vừa làm việc với hệ điều hành chủ, vừa có thể “làm việc” với hệ điều hành khác. Vì KIT KM9260 chạy trên hệ điều hành Linux nên các chương trình ứng dụng và các Driver mà ta viết ra phải dùng hệ điều hành Linux biên dịch.

Điểm mới ở VMware Workstation là có thể copy tài liệu qua lại giữa máy thật và máy ảo, hỗ trợ cổng USB 2.0, hiển thị các HĐH ảo qua nhiều Tab trên một cửa sổ (Tab giống như trình duyệt Web).

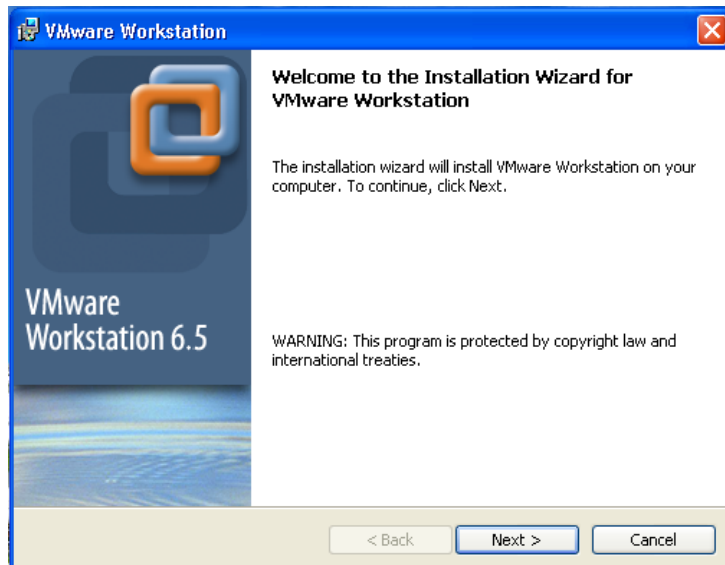
Hiện nay VMware Workstation và Linux có rất nhiều phiên bản khác nhau, nhưng ở đây chúng ta dùng **VMware Workstation v6.5** và **Linux UBUNTU v11.04**.

II. Các bước cài đặt VMware Workstation v6.5:

VMware giúp giả lập máy tính ảo trên một máy tính thật. Sau khi cài đặt VMware, ta có thể tạo các máy ảo chia sẻ CPU, RAM, Card mạng với máy tính thật. Điều này cho phép xây dựng nên một hệ thống với một vài máy tính được nối với nhau theo một mô hình nhất định, người sử dụng có thể tạo nên hệ thống của riêng mình, cấu hình theo yêu cầu của bài học.

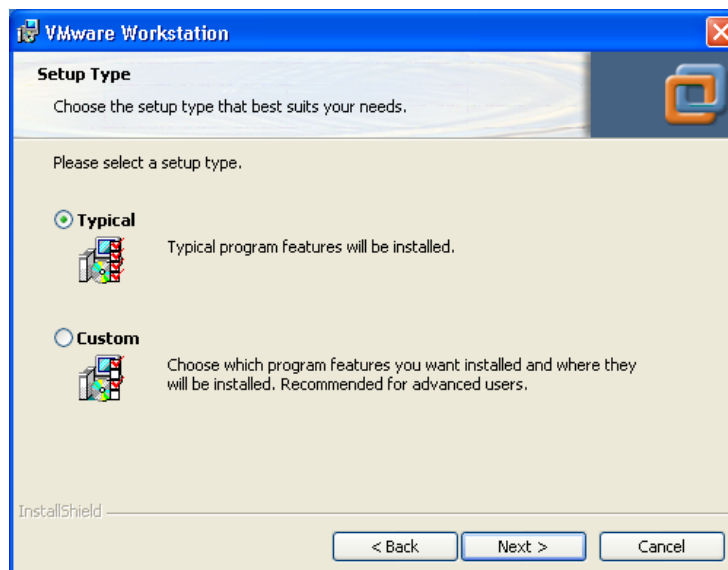
Cách cài đặt VMware Workstation 6.5 giống như các phần mềm khác:

- Chạy tập tin **VMware-workstation.exe**. Hộp thoại Welcome xuất hiện.
Chọn **Next** để tiếp tục



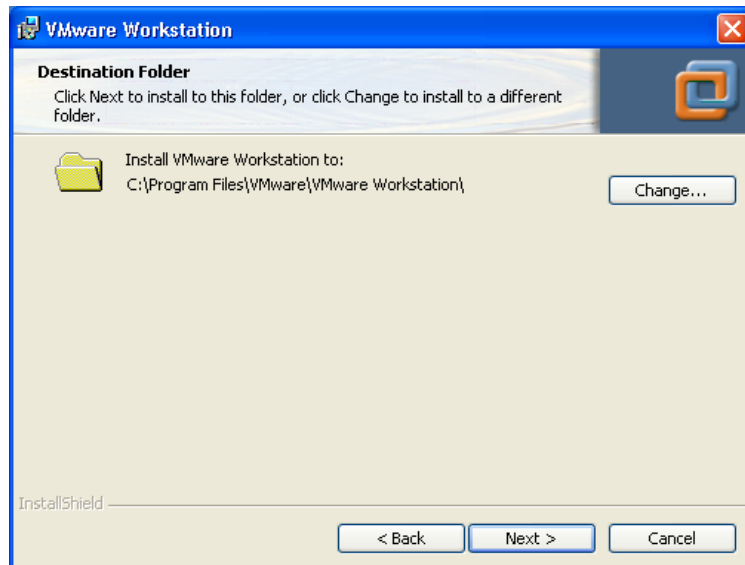
Hình 2.43: Bước 1 cài đặt phần mềm VMware-workstation

- Hộp thoại Setup Type xuất hiện. Chọn chế độ cài đặt **Typical** (sử dụng các thiết lập mặc định). Chọn **Next** để tiếp tục



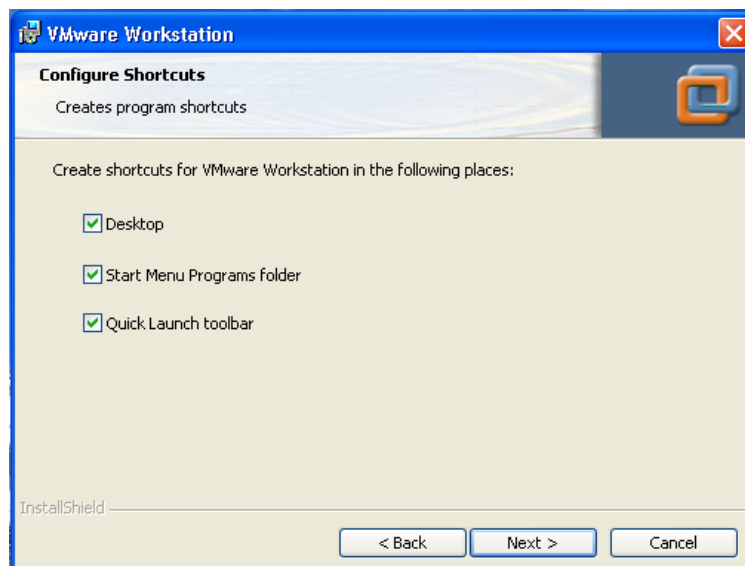
Hình 2.44: Bước 2 cài đặt phần mềm VMware-workstation

- Hộp thoại Destination Folder cho biết thư mục cài đặt chương trình. Có thể chọn thư mục cài đặt khác bằng cách nhấp chọn nút Change.... Chọn **Next** để tiếp tục



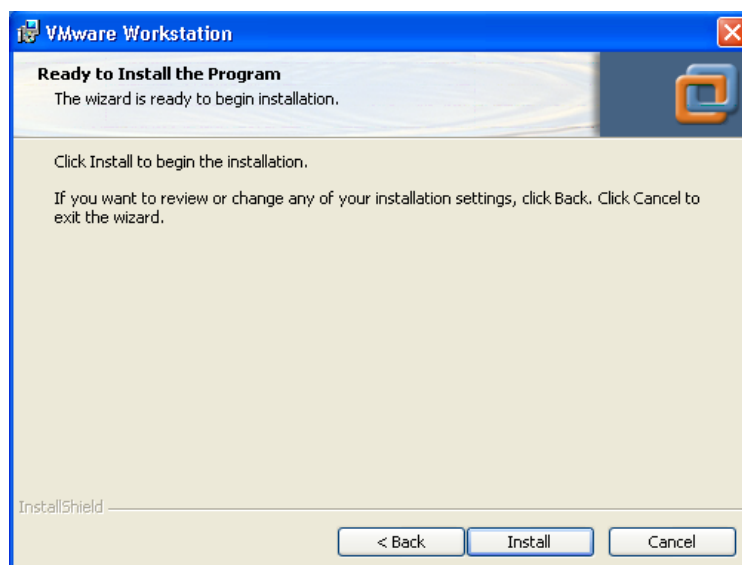
Hình 2.45: Bước 3 cài đặt phần mềm VMware-workstation

- Hộp thoại Configure Shortcuts cho phép tạo Shortcut chương trình trên (**Desktop**: trên màn hình Desktop, **Start Menu**: trong Menu Start, **Quick Launch**: trên thanh khởi động nhanh). Chọn **Next** để tiếp tục



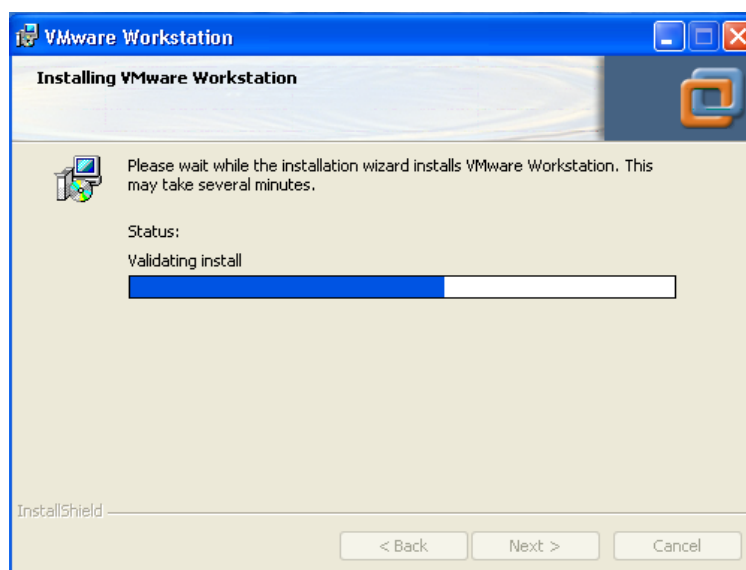
Hình 2.46: Bước 4 cài đặt phần mềm VMware-workstation

- Chọn **Install** để tiến hành cài đặt chương trình



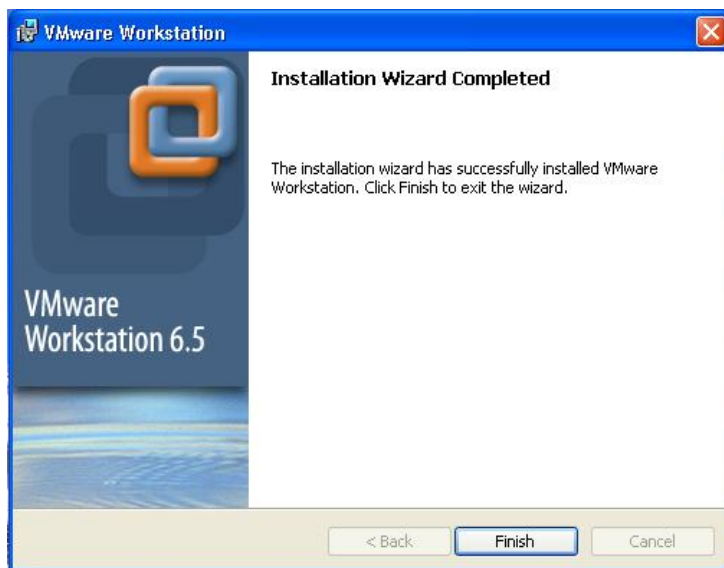
Hình 2.47: Bước 5 cài đặt phần mềm VMware-workstation

- Chúng ta đợi đến khi nào chương trình cài đặt xong.



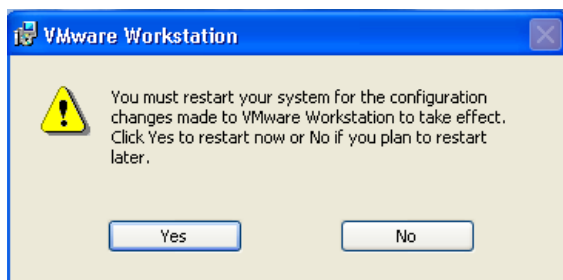
Hình 2.48: Bước 6 cài đặt phần mềm VMware-workstation

- Hộp thoại thông báo đã hoàn tất quá trình cài đặt xuất hiện. Chúng ta nhấn Finish để hoàn thành quá trình cài đặt.



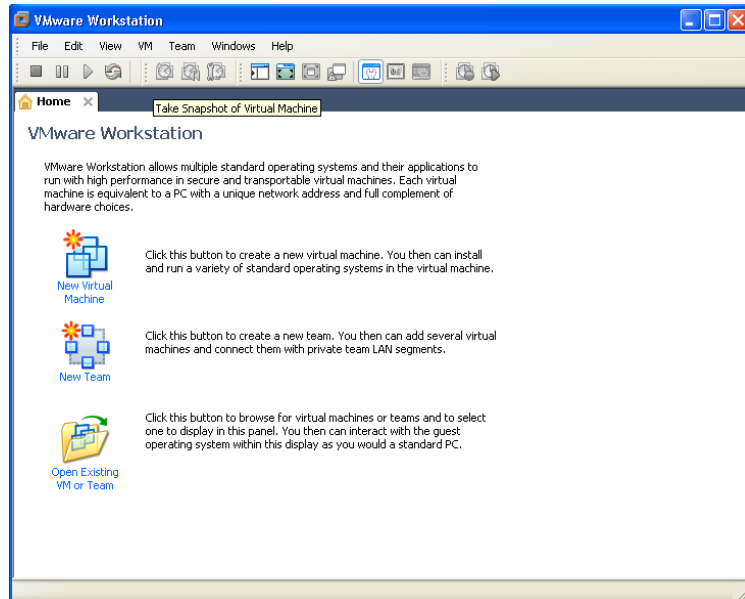
Hình 2.49: Bước 7 cài đặt phần mềm VMware-workstation

- Một hộp thoại khác hiện ra yêu cầu chúng ta phải khởi động lại máy tính. Chúng ta nhấn Yes



Hình 2.50: Bước 8 cài đặt phần mềm VMware-workstation

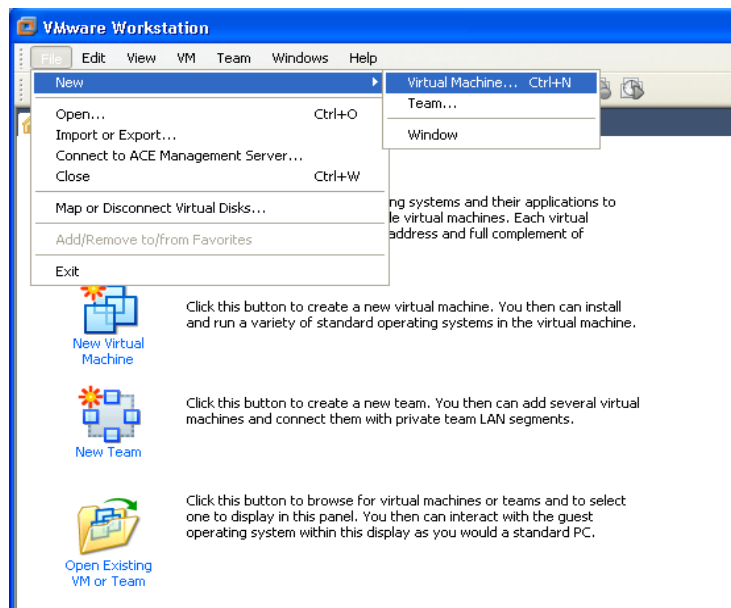
- Sau khi máy tính khởi động lại chúng ta click đôi vào Icon của VMware Workstation. Màn hình chính của chương trình hiện ra



*Hình 2.51:
Bước 9 cài đặt phần mềm
VMware-workstation*

III. Các bước cài đặt máy tính ảo bằng phần mềm VMware Workstation v6.5:

- Mở phần mềm VMware Workstation, ta được giao diện chính của chương trình. Chúng ta click vào menu File > New > Virtual Machine để tạo mới một máy ảo.



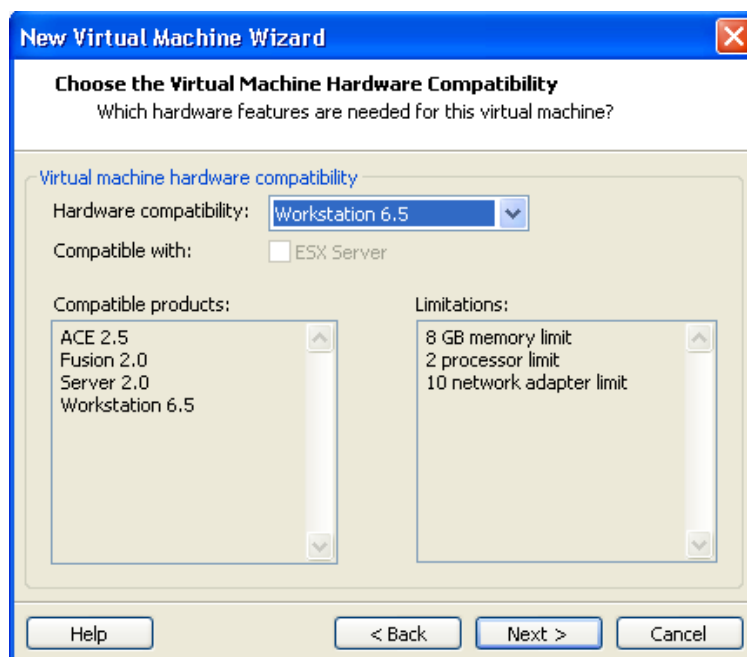
Hình 2.52: Bước 1 cài đặt máy tính ảo bằng VMware-workstation

- Cửa sổ chào mừng xuất hiện có hai sự lựa chọn:
 - + Typical(recommended) : tiêu biểu (được khuyên dùng) cấu hình mặc định.
 - + Custom (advanced): tùy chỉnh (nâng cao) cấu hình tùy chỉnh theo ý muốn.
 - + Ở đây chúng ta chọn Custom để dễ thiết lập cấu hình chọn xong bạn nhấn Next.



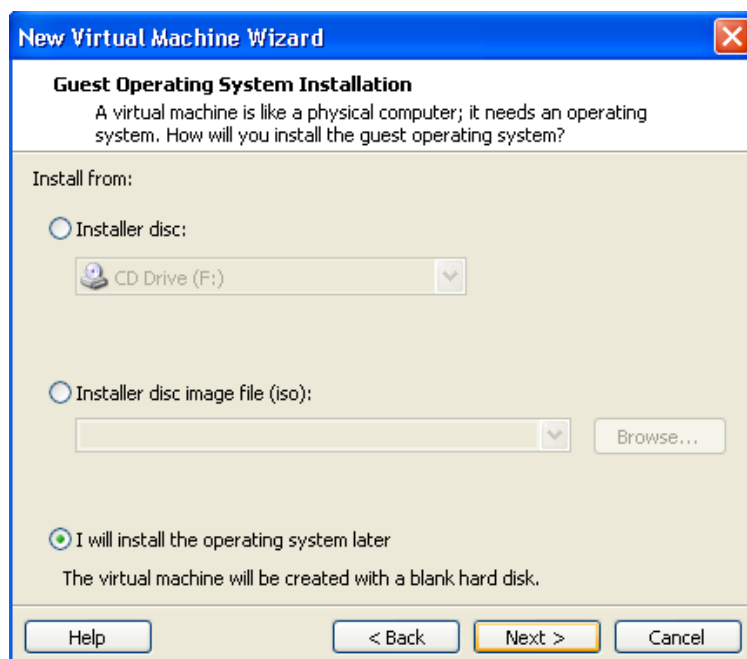
Hình 2.53: Bước 2 cài đặt máy tính ảo bằng VMware-workstation

- Trong cửa sổ tiếp theo mục Hardware compatibility bạn chọn Workstation 6.5. Chọn xong nhấn Next.



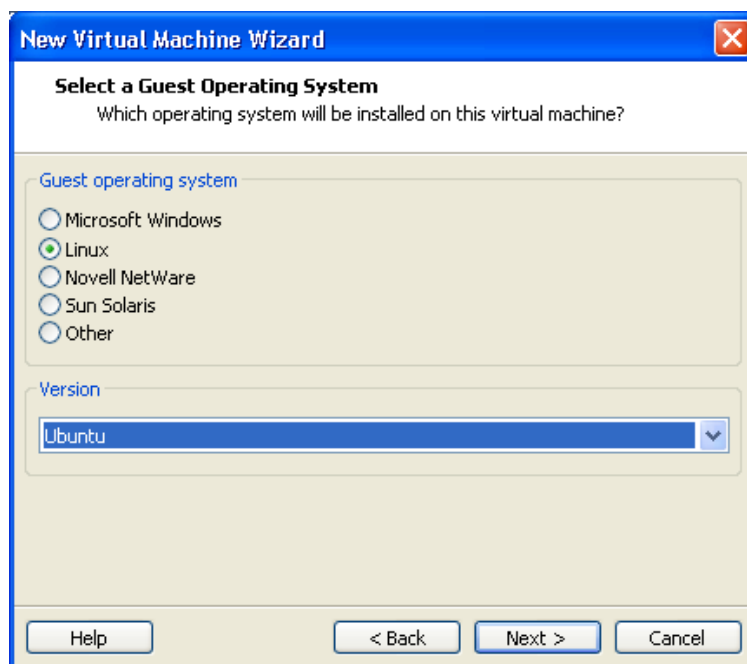
Hình 2.54: Bước 3 cài đặt máy tính ảo bằng VMware-workstation

- Tiếp theo bạn chọn I will install the operating system later – “Tôi sẽ cài đặt hệ điều hành sau”. Xong bạn nhấn Next.



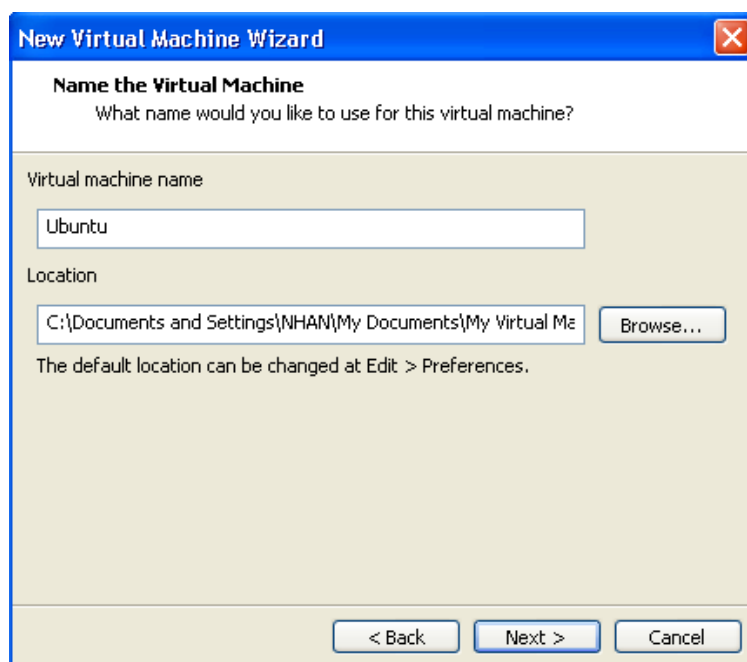
Hình 2.55: Bước 4 cài đặt máy tính ảo bằng VMware-workstation

- Tiếp theo là chọn hệ điều hành và phiên bản của hệ điều hành đó, mục Guest operating system chọn Linux, mục Version chọn Ubuntu.



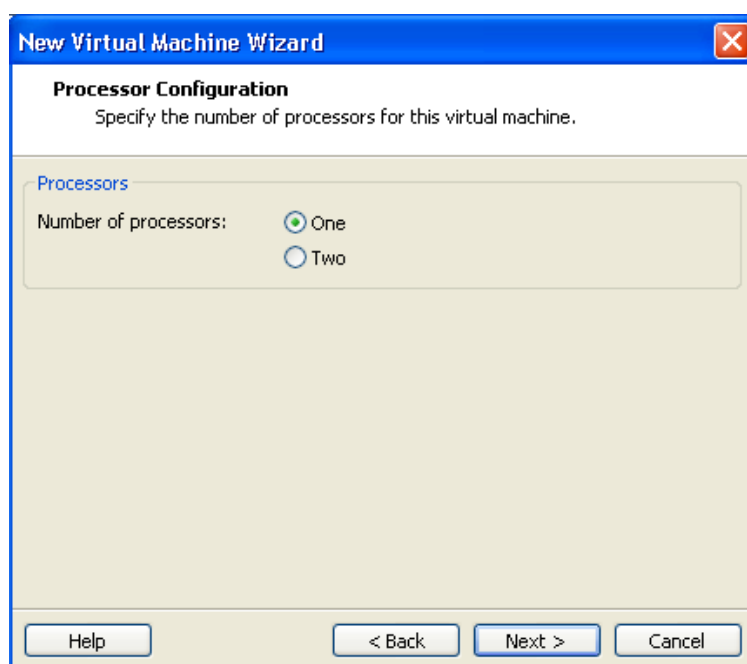
Hình 2.56: Bước 5 cài đặt máy tính ảo bằng VMware-workstation

- Đặt tên cho máy ảo và chọn nơi lưu.



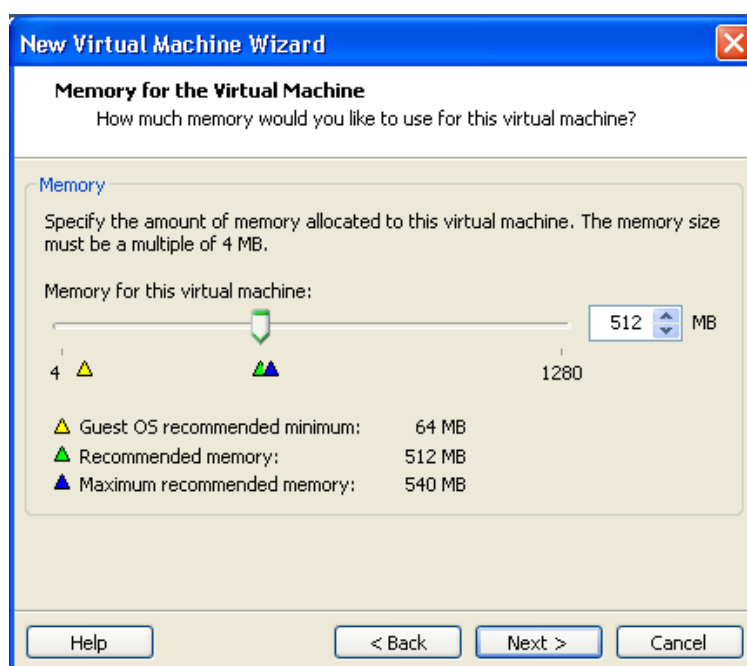
Hình 2.57: Bước 6 cài đặt máy tính ảo bằng VMware-workstation

- Chọn số bộ vi xử lý (Number of processors) cho máy ảo. Xong nhấn Next.



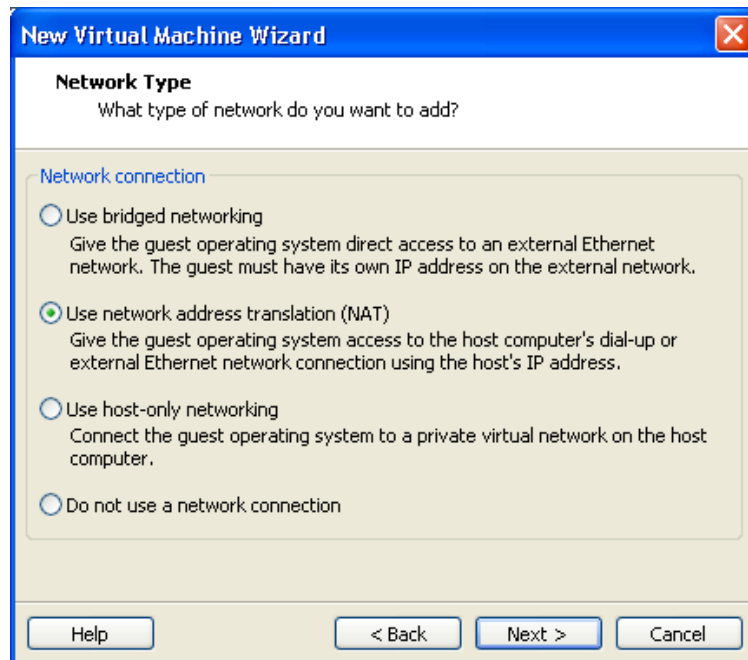
Hình 2.58: Bước 7 cài đặt máy tính ảo bằng VMware-workstation

- Chọn bộ nhớ chính (RAM) cho máy ảo mặc định là 512MB. Chọn xong nhấn Next.



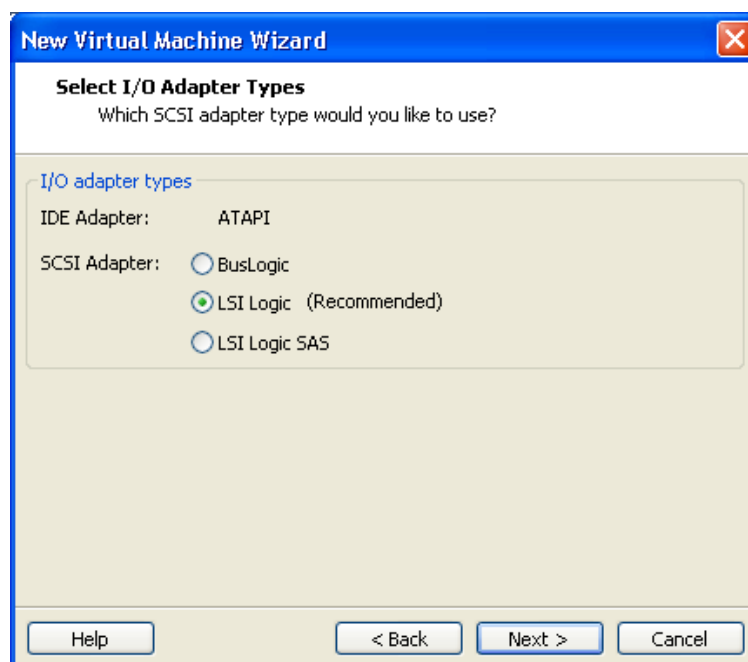
Hình 2.59: Bước 8 cài đặt máy tính ảo bằng VMware-workstation

- Chọn kiểu kết nối mạng chọn Use network address translation (NAT) và nhấn Next.



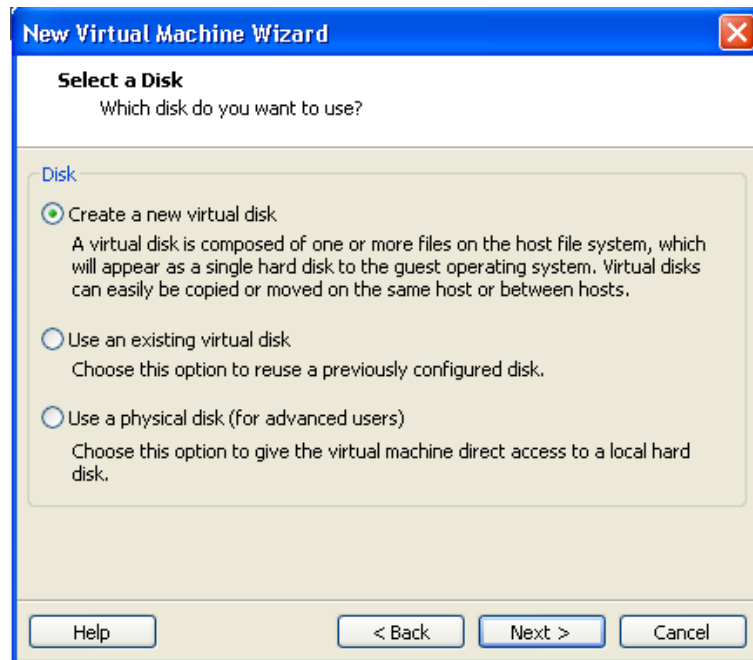
Hình 2.60: Bước 9 cài đặt máy tính ảo bằng VMware-workstation

- Mục Select I/O Adapter Types chọn LSI Logic (Recommended) nhấn Next.



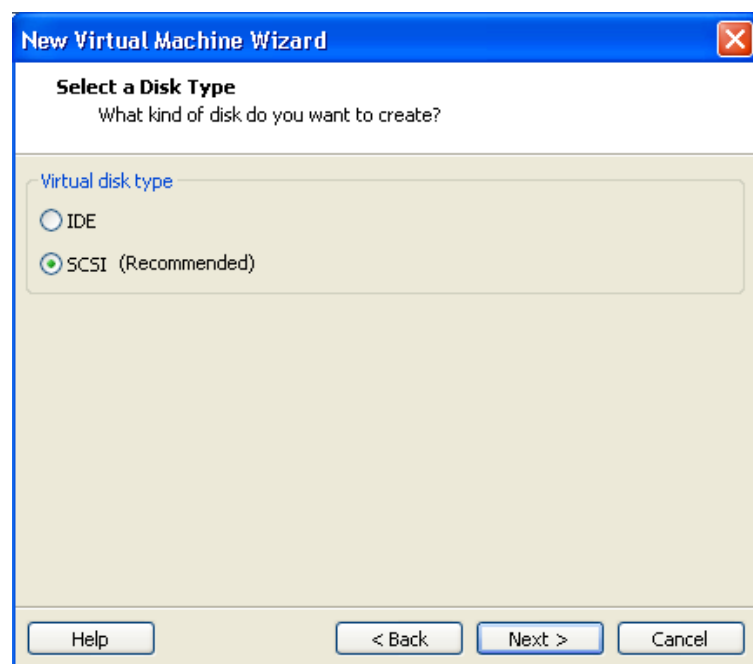
Hình 2.61: Bước 10 cài đặt máy tính ảo bằng VMware-workstation

- Mục Disk chọn Create a new virtual disk để tạo mới một ổ đĩa ảo để cài hệ điều hành chọn xong nhấn Next.



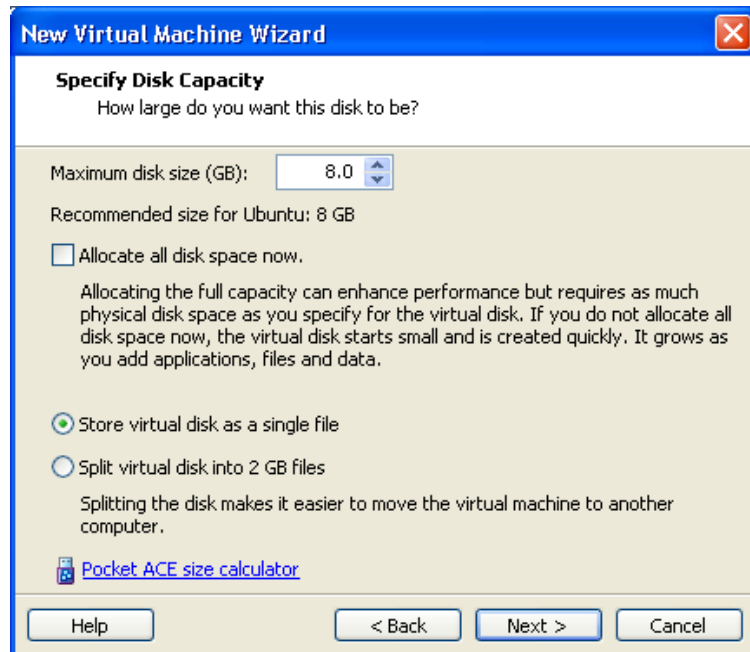
Hình 2.62: Bước 11 cài đặt máy tính ảo bằng VMware-workstation

- Chọn kiểu giao tiếp cho ổ đĩa ảo chọn SCSI (Recommended) và nhấn Next.



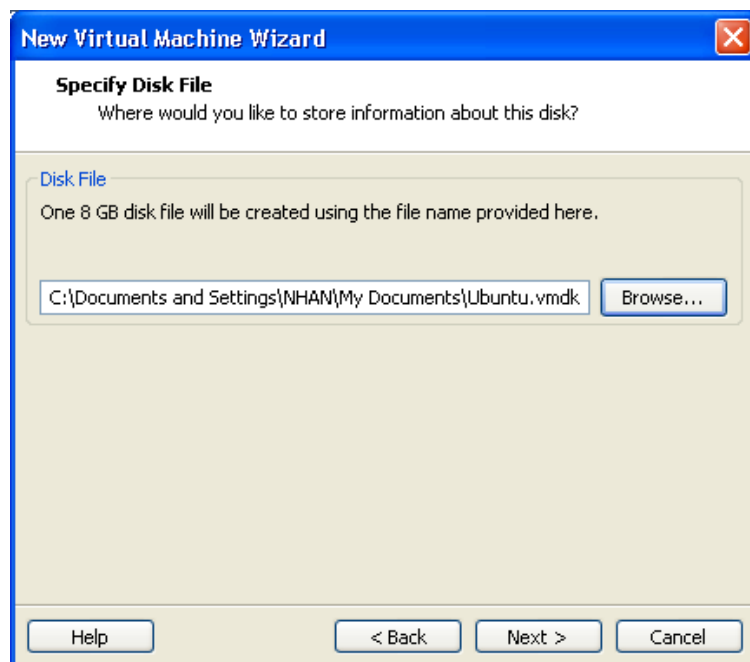
Hình 2.63: Bước 12 cài đặt máy tính ảo bằng VMware-workstation

- Mục Maximum disk size(GB) chọn dung lượng tối đa cho ổ đĩa ảo mặc định là 8.0 GB chọn tiếp mục Store virtual disk as a single file xong nhấn Next.



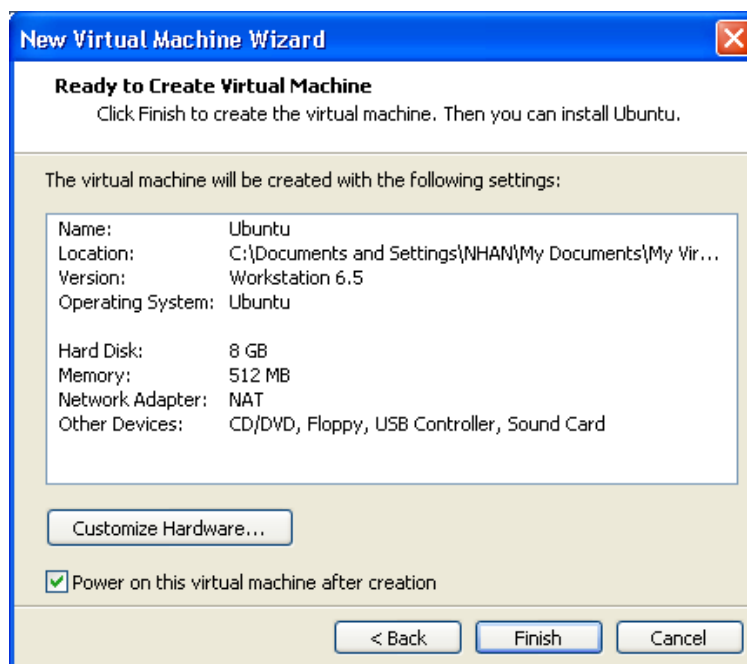
Hình 2.64: Bước 13 cài đặt máy tính ảo bằng VMware-workstation

- Tiếp theo là chọn nơi lưu trữ ổ đĩa ảo.



Hình 2.65: Bước 14 cài đặt máy tính ảo bằng VMware-workstation

- Cửa sổ thông báo Ready to Create Virtual Machine (sẵn sàng tạo máy ảo) xuất hiện bạn chọn Finish để hoàn tất quá trình tạo máy ảo.



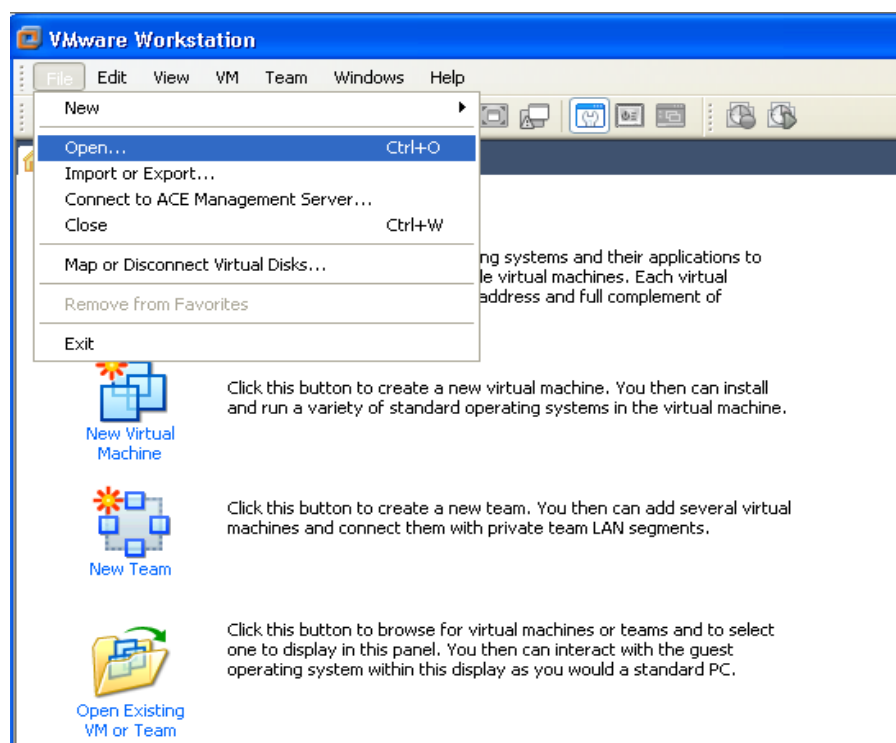
Hình 2.66: Bước 15 cài đặt máy tính ảo bằng VMware-workstation

Như vậy là máy tính ảo đã được tạo. Chúng ta khởi động máy ảo bằng cách nhấn vào Power on this virtual machine. Chúng ta có thể thấy quá trình khởi động máy ảo giống như quá trình khởi động của máy thật.

Chúng ta có thể cài đặt hệ điều hành cho máy ảo bằng cách cho đĩa cài đặt vào ổ CD-ROM và khởi động máy ảo. Quá trình cài đặt hoàn toàn giống máy thật.

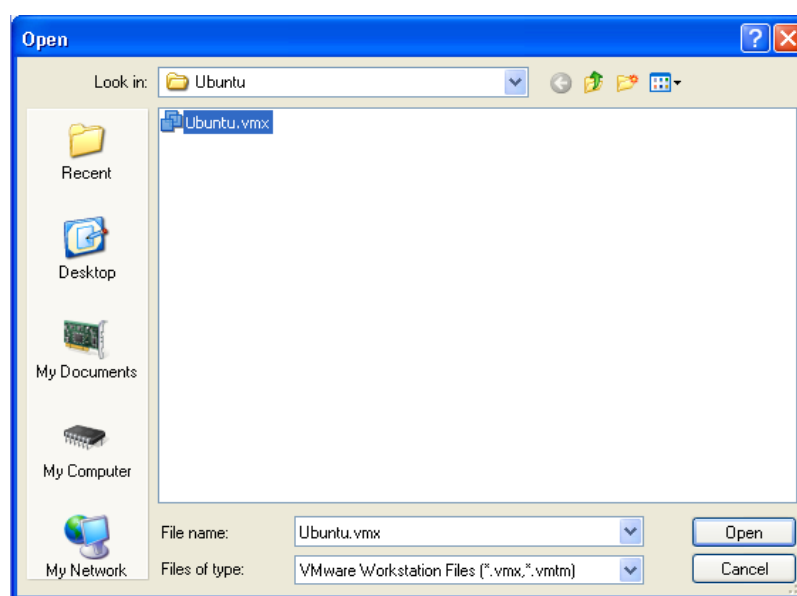
Tuy nhiên vì việc cài hệ điều hành tương đối mất thời gian và phức tạp nên trong thực tế người ta thường tạo ra máy ảo cài đặt sẵn hệ điều hành và lưu trữ dưới dạng thư mục do VMWARE WORKSTATION tạo ra trong quá trình tạo máy ảo và cài đặt hệ điều hành. Chúng ta chỉ cần chạy file *.vmx trong thư mục này bằng phần mềm VMWARE WORKSTATION v6.5 là chúng ta đã có máy ảo có cài đặt sẵn hệ điều hành mà chúng ta mong muốn. Ở đây chúng ta có file Ubuntu.vmx trong thư mục Ubuntu11.04. Chúng ta sẽ chạy máy ảo này như sau:

- Chọn File > Open



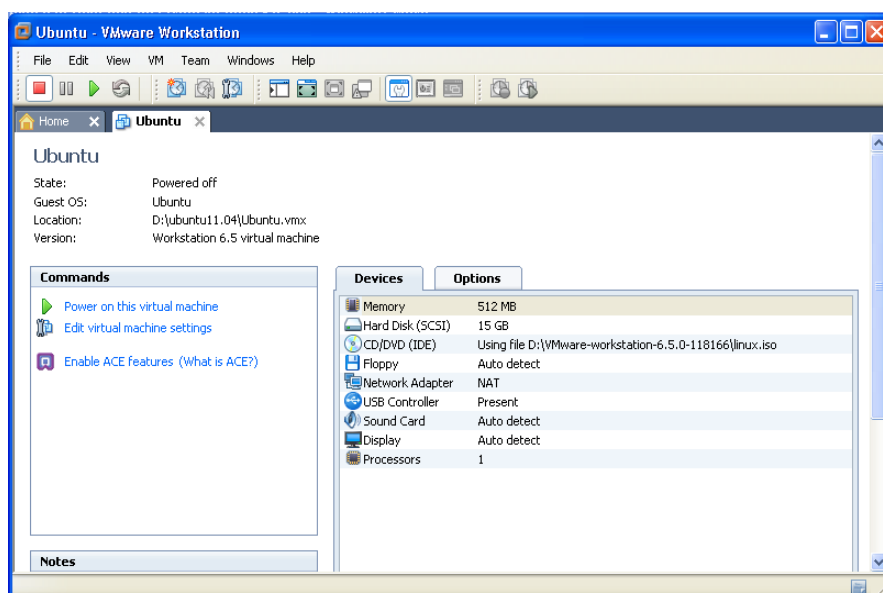
Hình 2.67: Bước 1 mở máy ảo cài đặt sẵn

- Một hộp thoại hiện ra yêu cầu ta chọn file *.vmx để chạy. Chúng ta dẫn đến thư mục Ubuntu như đã nói trên và chọn Ubuntu.vmx. Xong chọn Open



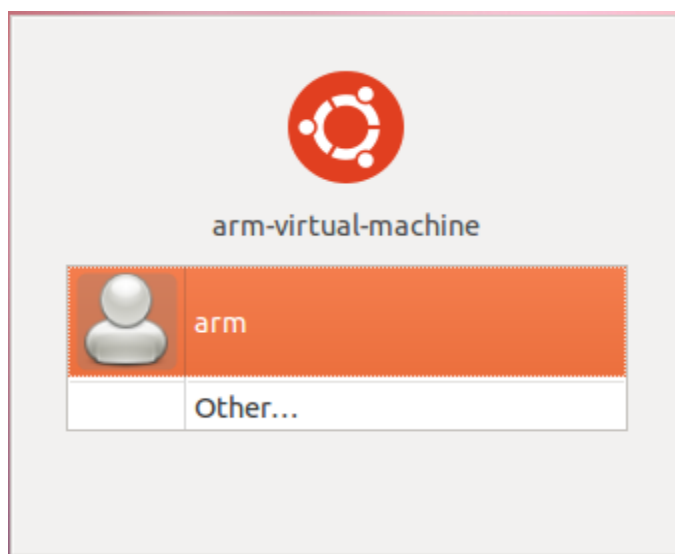
Hình 2.68: Bước 2 mở máy ảo cài đặt sẵn

- Chúng ta nhấn vào Power on this virtual machine để bắt đầu chạy máy ảo này.



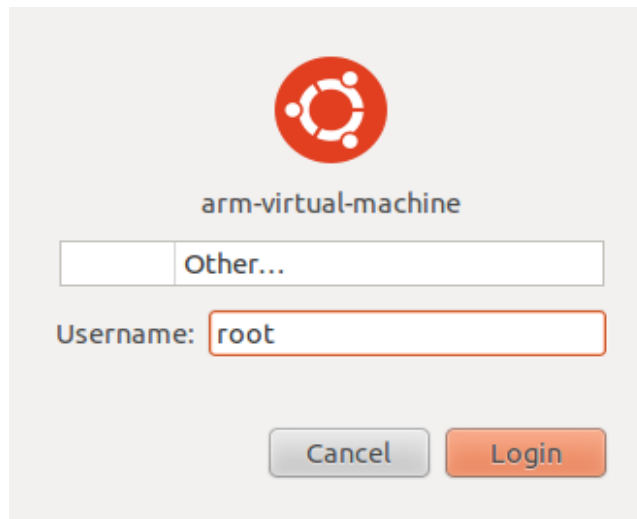
Hình 2.69: Bước 3 mở máy ảo cài đặt sẵn

- Máy tính ảo bắt đầu chạy. Khi máy tính ảo chạy đến phần chọn tài khoản người dùng thì chúng ta chọn Other để tạo một tài khoản mới

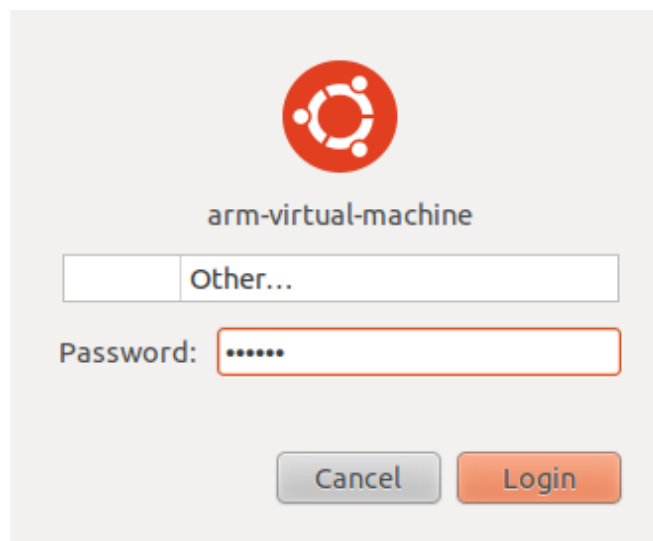


Hình 2.70: Bước 4 mở máy ảo cài đặt sẵn

- Ở đây chúng ta dùng tên tài khoản (User name) là root và mật khẩu (Password) là root00.



Hình 2.71: Bước 5 mở máy ảo cài đặt sẵn



Hình 2.72: Bước 6 mở máy ảo cài đặt sẵn

Như vậy là chúng ta đã hoàn tất quá trình tạo máy tính ảo và cài đặt hệ điều hành Linux Ubuntu cho máy ảo. Chúng ta có thể sử dụng hệ điều hành Linux song song với hệ điều hành Window.

B- Chương trình SAMBA.

I. Giới thiệu:

SAMBA là chương trình dùng để nạp dữ liệu lên các vùng nhớ trên KIT KM9260. Như chúng ta đã đọc ở các phần trước, sau quá trình Bootloader nếu vi xử lý không tìm thấy Bootstrapcode trên các phân vùng nhớ của KIT KM9260 thì vi xử lý sẽ nhảy sang chương trình SAMBA Boot. Khi ở chế độ SAMBA Boot thì vi xử lý cho phép ta được chép dữ liệu từ máy tính vào trong các vùng nhớ trên KIT KM9260 thông qua cổng USB Device bằng phần mềm SAMBA

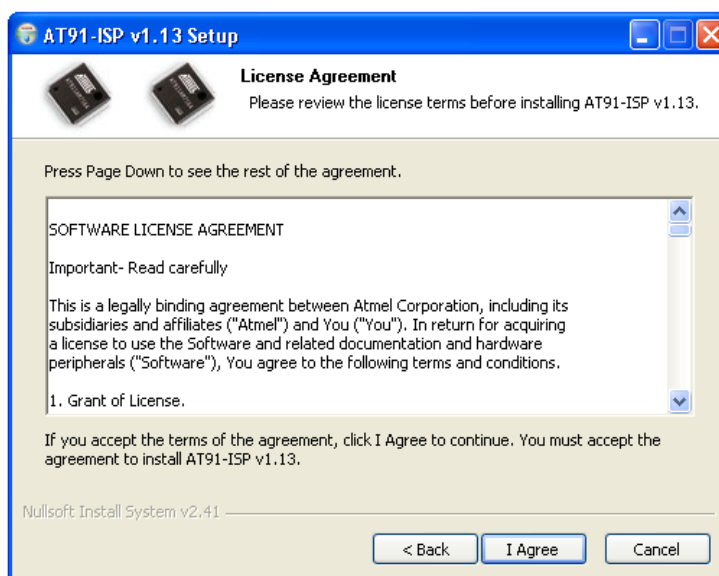
II. Quá trình cài đặt Samba:

- File cài đặt chương trình SAMBA có tên là Install AT91-ISP v1.13.exe. Ta chạy file này.
- Hộp thoại AT91-ISP v1.13 Setup hiện ra, ta click Next để tiếp tục cài đặt



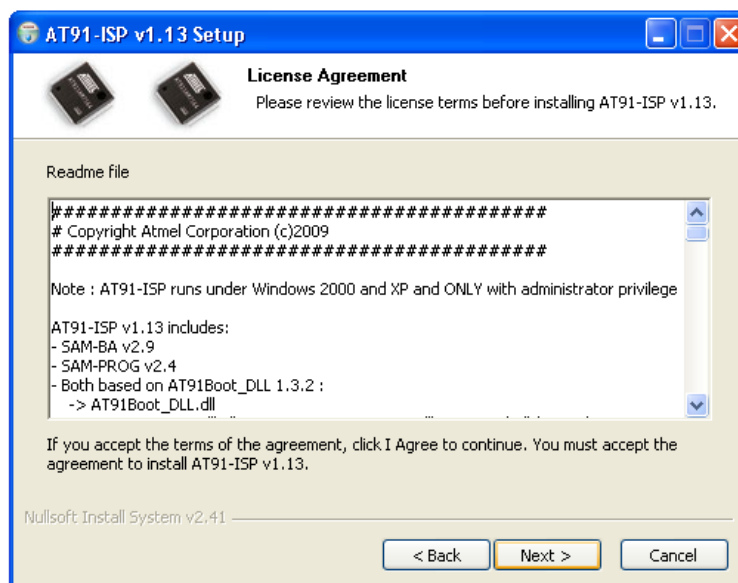
Hình 2.73: Bước 1 cài đặt SAMBA

- Hộp thoại tiếp theo hiện ra, bạn chọn Agree để đi tiếp.



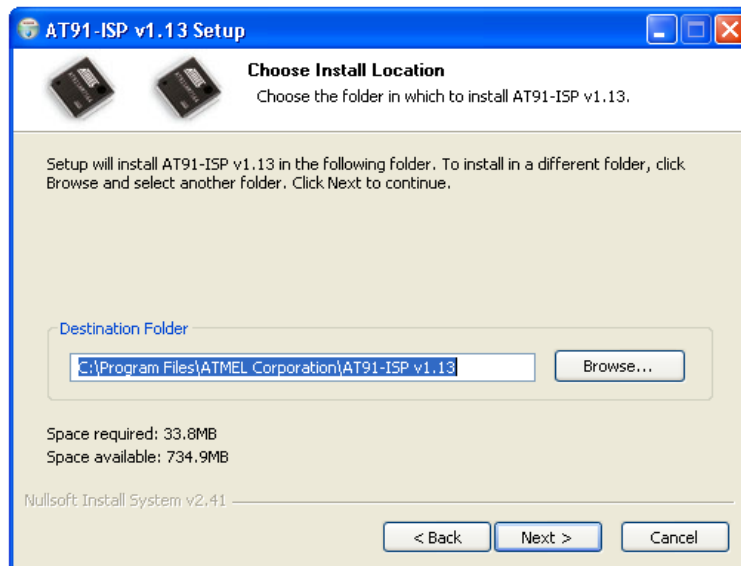
Hình 2.74: Bước 2 cài đặt SAMBA

- Tiếp theo bạn nhấn Next.



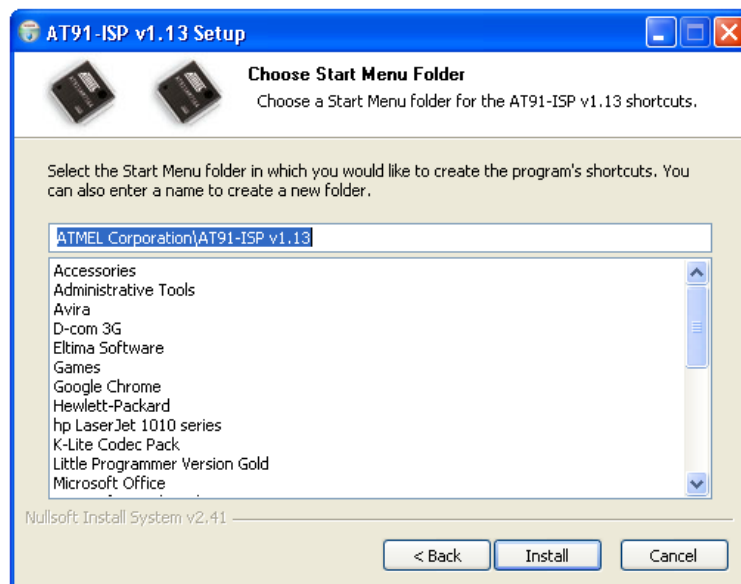
Hình 2.75: Bước 3 cài đặt SAMBA

- Hộp thoại kế tiếp hiện ra, yêu cầu bạn chọn nơi lưu chương trình. Mặc định là C:\Program Files\ATMEL Corporation\AT91-ISP v1.13, nhưng bạn có thể thay đổi vị trí lưu chương trình bằng cách click vào Browse và chọn nơi cần lưu chương trình. Chọn xong ta nhấn Next



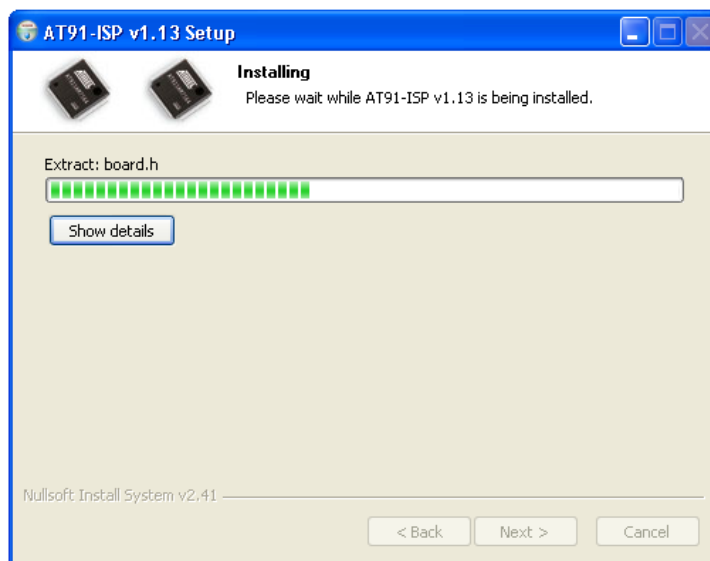
Hình 2.76: Bước 4 cài đặt SAMBA

- Hộp thoại hiện ra yêu cầu ta chọn thư mục chứa Shortcut, mặc định là ATMEL Corporation\AT91-ISP v1.13. Chọn xong ta click Install để cài đặt.



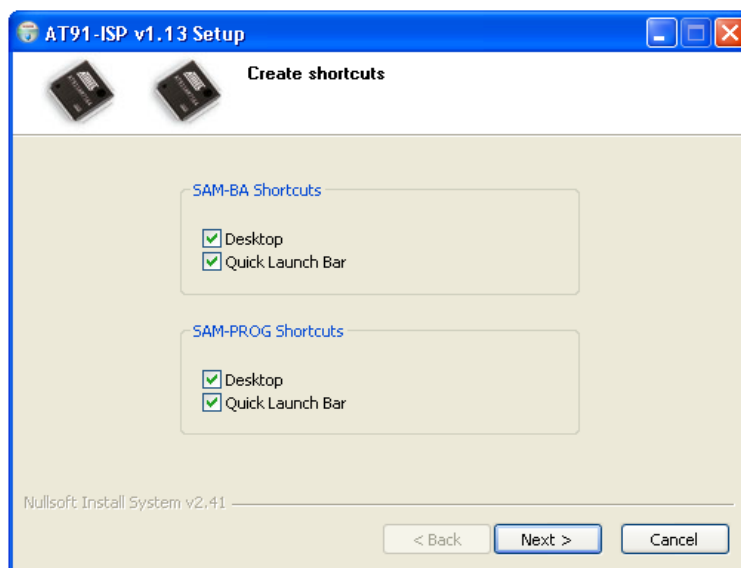
Hình 2.77: Bước 5 cài đặt SAMBA

- Tiếp theo sẽ hiện ra hộp thoại cài đặt. Ta chờ đến khi nào chương trình cài đặt xong thì ta nhấn Next



Hình 2.78: Bước 6 cài đặt SAMBA

- Hộp thoại tiếp theo yêu cầu ta có đặt Shortcut ra Desktop và chạy SAMBA luôn không. Chọn xong ra nhấn Next.



Hình 2.79: Bước 7 cài đặt SAMBA

- Hộp thoại cuối cùng hiện ra báo đã cài đặt thành công và yêu cầu khởi động lại máy tính. Bạn click Finish.



Hình 2.80: Bước 8 cài đặt SAMBA

- Nếu chúng ta có chọn tạo Shortcut ở Desktop thì trên Desktop sẽ có Icon của SAMBA là:

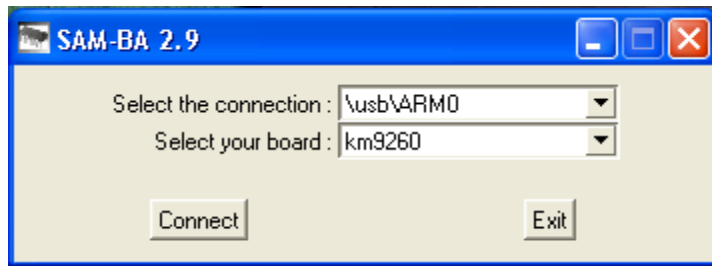


- Sau đó bạn chép thư mục AT91-ISP v1.13 đè lên thư mục cài đặt của chương trình C:\Program Files\ATMEL Corporation\AT91-ISP v1.13.

III. Hướng dẫn sử dụng Samba:

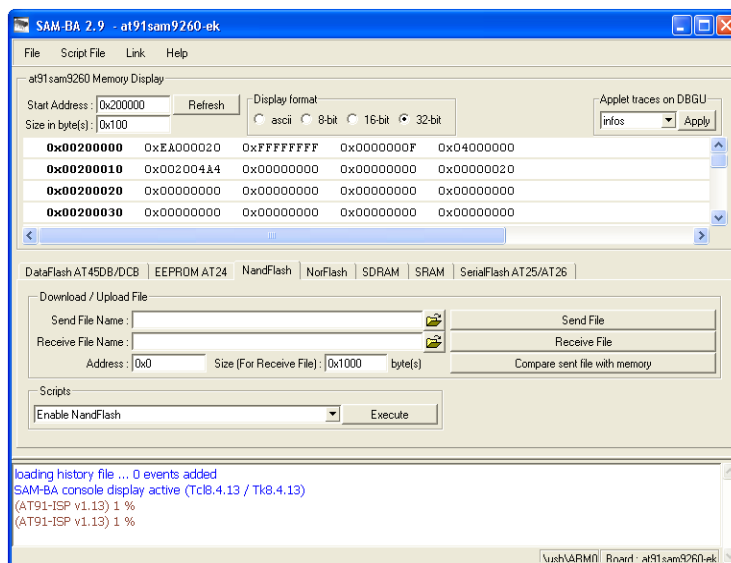
- Trước tiên, ta kết nối KIT KM9260 với máy tính thông qua cổng USB Devices.
- Sau đó cách ly vi xử lý với các vùng nhớ chứa Bootstrapcode bằng cách tháo Jump15 trên KIT ra, đây là Jump nối Serial DataFlash với vi xử lý. Theo phân vùng bộ nhớ của KIT KM9260 thì Serial DataFlash là nơi chứa Bootstrapcode cần thiết cho quá trình khởi động của KIT, khi ta tháo Jump15 ra thì trong quá trình Bootloader vi xử lý sẽ không tìm thấy Bootstrapcode và chuyển sang chế độ Boot SAMBA. Lúc này trên máy tính sẽ nhận diện KIT KM9260 như là một thiết bị ngoại vi.
- Cấp nguồn cho KIT để KIT hoạt động.
- Sau khi máy tính đã nhận ra KIT KM9260 thì ta nối Jump15 lại để có thể nạp dữ liệu vào Serial DataFlash.

- Khi Click đôi vào Icon của SAMBA thì sẽ hiện ra hộp thoại yêu cầu chọn kiểu kết nối và loại KIT mà chúng ta dùng. Ta chọn như hình sau và nhấn Connect



Hình 2.81: Hộp thoại chọn kiểu và loại kit kết nối của SAMBA

- Sau đó giao diện chính của SAMBA sẽ hiện ra

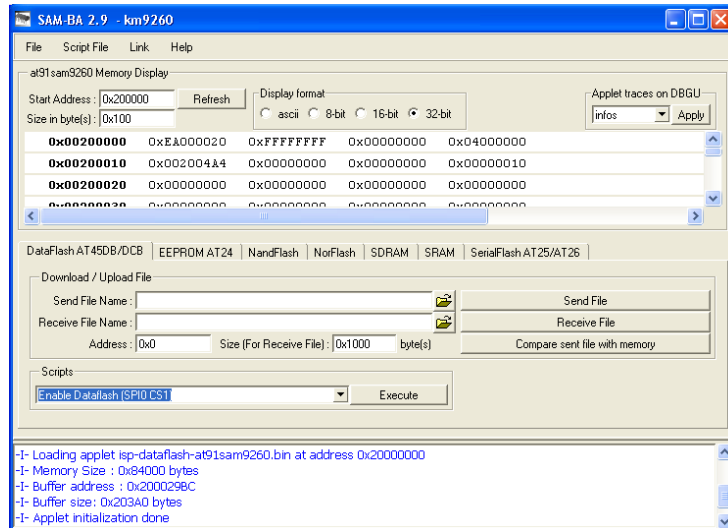


Hình 2.82: Giao diện chính của SAMBA

- Trên giao diện chính của SAMBA ta thấy có rất nhiều thẻ ứng với việc nạp vào các vùng nhớ khác nhau. Nhưng ở đây chúng ta chỉ quan tâm đến việc nạp dữ liệu vào vùng nhớ DataFlash và NandFlash nên ta chỉ quan tâm đến thẻ DataFlashAT45DB/DCB và NandFlash.
- Việc nạp File gì và địa chỉ offset của File lên các chip nhớ là do phân vùng bộ nhớ của KIT KM9260 quy định. Như đã trình bày ở các phần trước thì KIT KM9260 có hai phân vùng bộ nhớ là phân vùng loại 1 và phân vùng loại 2.


1. Nạp code cho Dataflash:


- Chọn Tab là DataFlash AT45DB/DCB
- Chọn Scripts: Enable DataFlash (SPI0 CS1)
- Chọn nút Excute để cho phép truy xuất vào DataFlash
- Nếu làm đúng các thao tác trên thì sẽ có hình sau



Hình 2.83: Giao diện của SAMBA khi truy xuất DataFlash thành công

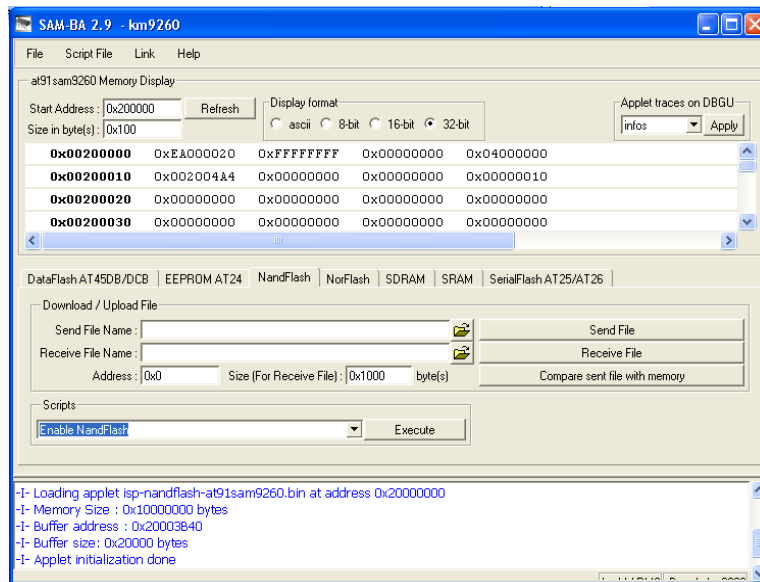
Sau khi hoàn thành 3 bước trên thì cho phép thực hiện tiếp công việc sau:

- Nạp BootstrapCode:
 - + Tại Scripts chọn: Send Boot File
 - + Tại Address chọn: địa chỉ offset của bootstrapcode trên DataFlash
 - + Nhấn nút Excute SW sẽ hiện ra một hộp thoại để chọn file bootstrap.bin là file bootstrapcode của KIT. Sau đó nhấn Open thì bootstrapcode sẽ tự động nạp vào dataflash.
- Nạp U-boot:
 - + Tại Scripts chọn Enable DataFlash (SPI0 CS1)
 - + Tại Address chọn địa chỉ offset của U-Boot trên DataFlash
 - + Tại Send File Name: Click vào biểu tượng  và chọn đường dẫn đến file U-Boot.bin.
 - + Nhấn nút Send File thì U-Boot sẽ được nạp vào DataFlash
- Xóa biến môi trường U-boot :
 - + Tại Scripts chọn Enable DataFlash (SPI0 CS1)
 - + Tại Address chọn địa chỉ offset của U-Boot Env trên Data Flash


- + Tại Send File Name: Click vào biểu tượng  và chọn đường dẫn đến file Clear_Env_0x4200.bin (nếu KIT có phân vùng bộ nhớ là loại 1)
- + Nhấn Send File thì các biến môi trường của U-Boot sẽ được xóa.
- Xóa toàn bộ Data Flash:
 - + Tại Scripts chọn Erase All
 - + Nhấn Excute sau đó chờ kết thúc.



2. Nạp code lên Nand Flash:

- Chọn Tab là NandFlash
- Chọn Scripts: Enable NandFlash
- Chọn nút Excute để cho phép truy xuất vào NandFlash
- Nếu làm đúng các thao tác trên thì sẽ có hình sau



Hình 2.84: Giao diện của SAMBA khi truy xuất NandFlash thành công

- Nạp BootstrapCode:
 - + Tại Scripts chọn: Send Boot File
 - + Tại Address chọn: địa chỉ offset của bootstrapcode trên NandFlash
 - + Nhấn nút Excute SW sẽ hiện ra một hộp thoại để chọn file bootstrap.bin là file bootstrapcode của KIT. Sau đó nhấn Open thì bootstrapcode sẽ tự động nạp vào NandFlash.
- Nạp U-boot:
 - + Tại Scripts chọn Enable NandFlash
 - + Tại Address chọn địa chỉ offset của U-Boot trên NandFlash
 - + Tại Send File Name: Click vào biểu tượng  và chọn đường dẫn đến file U-Boot.bin.
 - + Nhấn nút Send File thì U-Boot sẽ được nạp vào NandFlash
- Nạp Kernel:
 - + Tại Scripts chọn Enable NandFlash
 - + Tại Address chọn địa chỉ offset của Kernel trên NandFlash

- + Tại Send File Name: Click vào biểu tượng  và chọn đường dẫn đến file: uImage.
- + Nhấn nút Send File thì Kernel sẽ được nạp vào NandFlash
- Nạp Rootfs:
 - + Tại Scripts chọn Enable NandFlash
 - + Tại Address chọn địa chỉ offset của Rootfs trên NandFlash
 - + Tại Send File Name: Click vào biểu tượng  và chọn đường dẫn đến file *.jffs2
 - + Nhấn nút Send File thì Rootfs sẽ được nạp vào NandFlash
- Xóa toàn bộ Nand Flash:
 - + Tại Scripts chọn Erase All
 - + Nhấn Excute sau đó chờ kết thúc.

C- Chương trình Putty

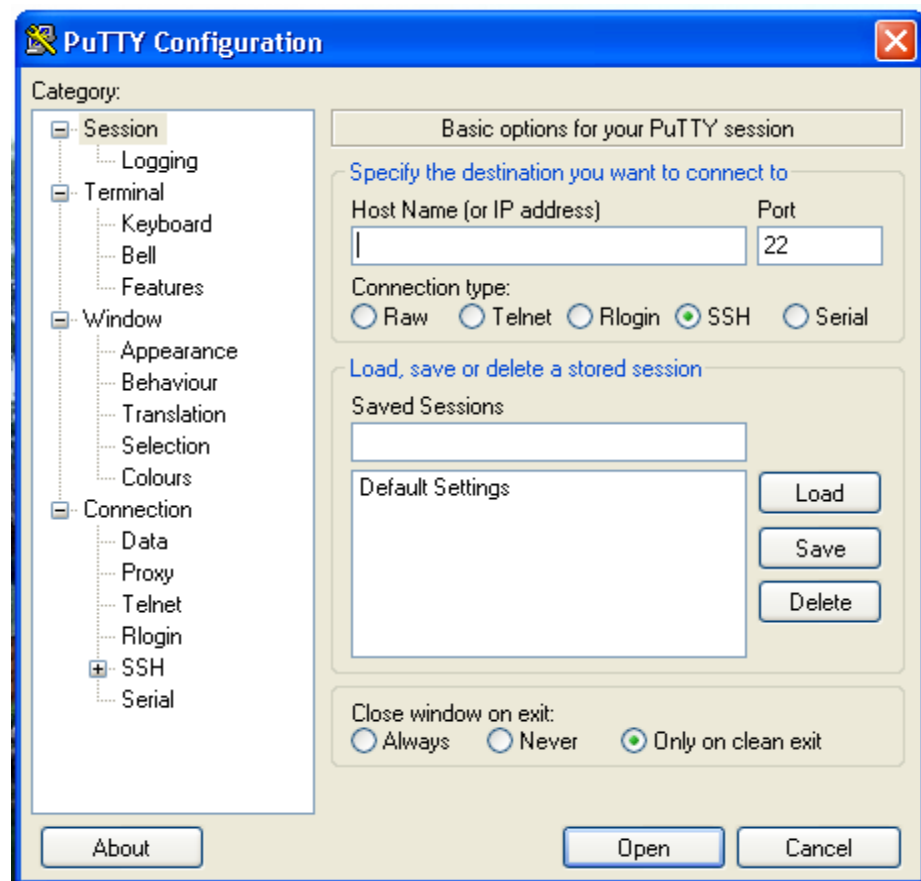
I. Giới thiệu:

Như đã giới thiệu ở phần trên người sử dụng thực hiện các thao tác chạy chương trình, viết chương trình,... trên KIT KM9260 phải thông qua màn hình Terminal của Linux. Nhưng KIT KM9260 không có hỗ trợ Card đồ họa cũng như màn hình nên ta phải dùng chương trình PUTTY để hiển thị màn hình Terminal của Linux trên màn hình máy tính.

Chương trình PUTTY kết nối với KIT KM9260 thông qua cổng truyền dữ liệu nối tiếp COM DB9.

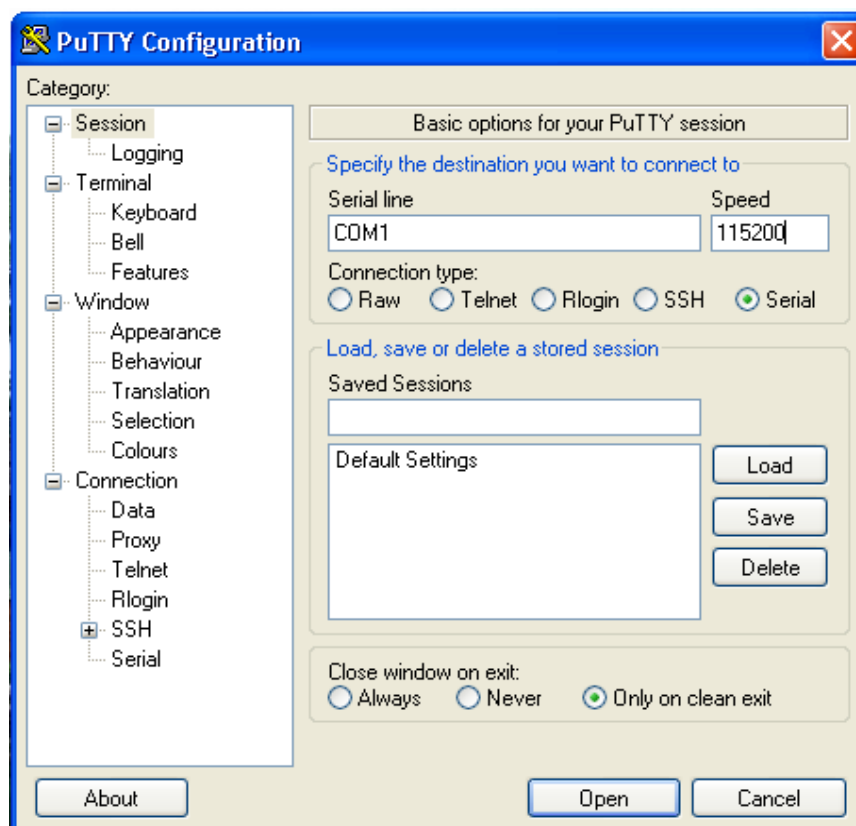
II. Hướng dẫn sử dụng Putty:

- Chương trình PUTTY là chương trình không cần cài đặt, người dùng chỉ cần chạy file putty.exe thì sẽ hiện ra giao diện



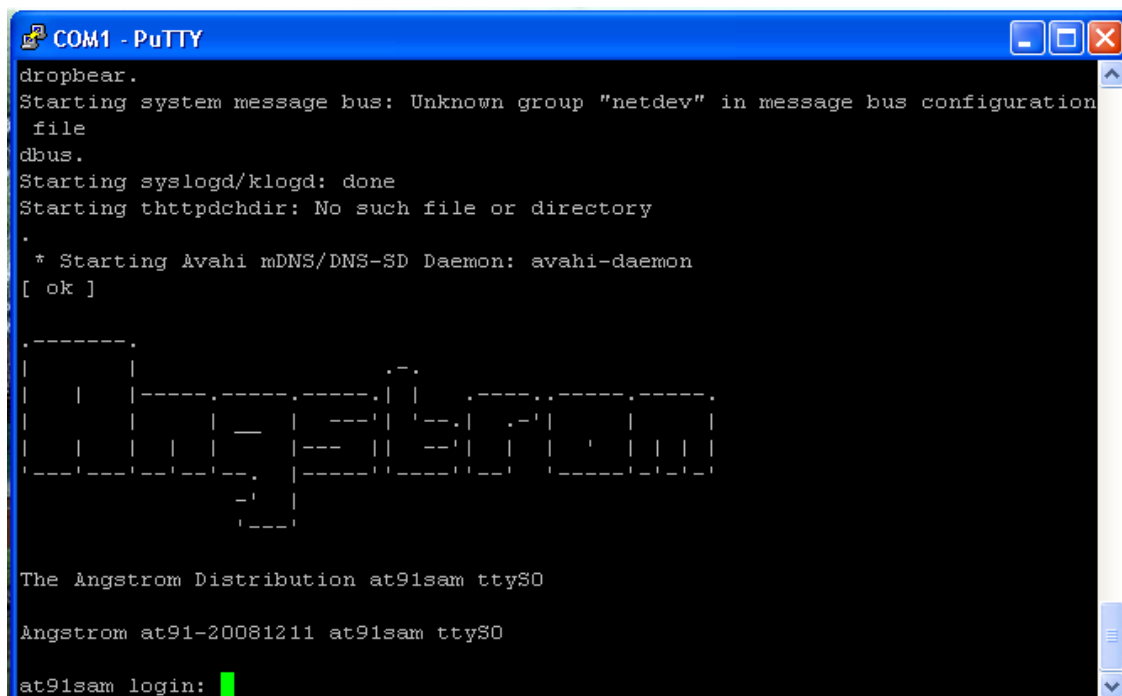
Hình 2.85: Giao diện chính của Putty

- Tại Specify the destination you want to connect to bạn chọn Serial và sửa Speed là 115200. Chọn xong nhấn Open



Hình 2.86: Nhập các thông số vào giao diện Putty

- Lúc này bạn sẽ thấy màn hình Terminal của Linux.



Hình 2.87: Giao diện của Linnux Angstron

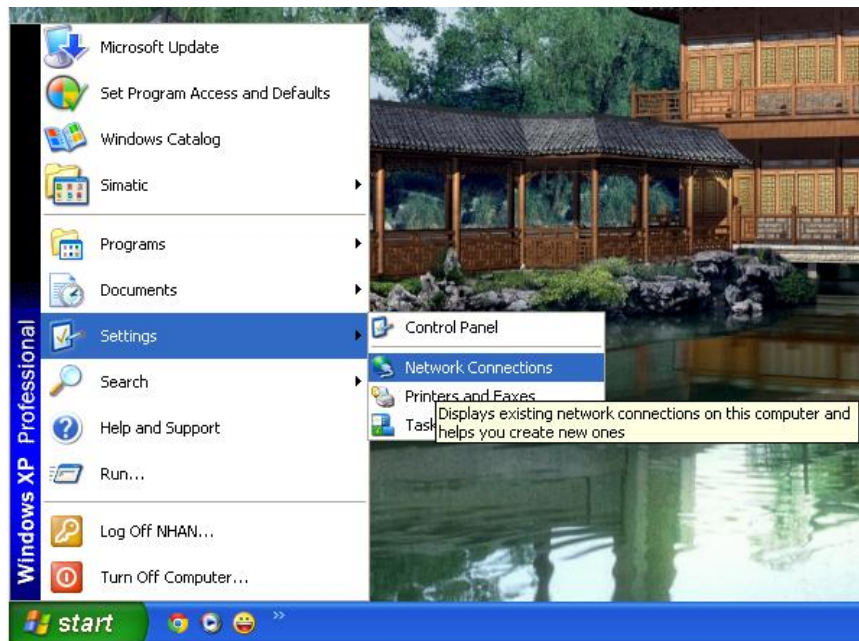
D- Chương trình tftpd32.

I. Giới thiệu:

- Tftpd32 là chương trình dùng để chép một file vào trong vùng nhớ của KIT.
- Tftpd32 giao tiếp với KIT KM9260 thông qua cổng Ethernet.

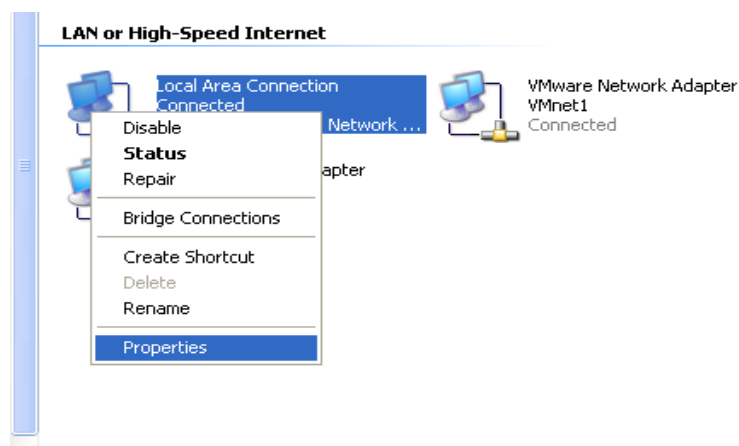
II. Hướng dẫn sử dụng tftpd32:

- Trước tiên ta cài đặt IP tĩnh cho máy tính. Bạn vào Start/Setting/Network Connections



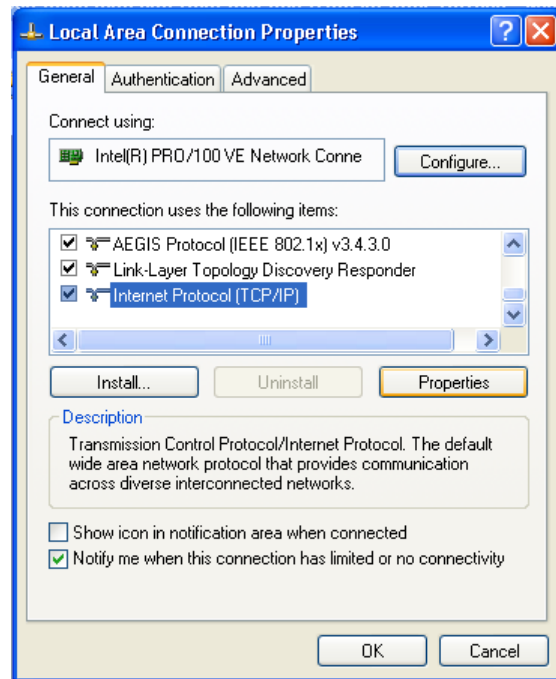
Hình 2.88: Bước 1 hướng dẫn sử dụng phần mềm tftpd32

- Một cửa sổ hiện ra bạn click chuột phải vào Local Area Connection và chọn Properties



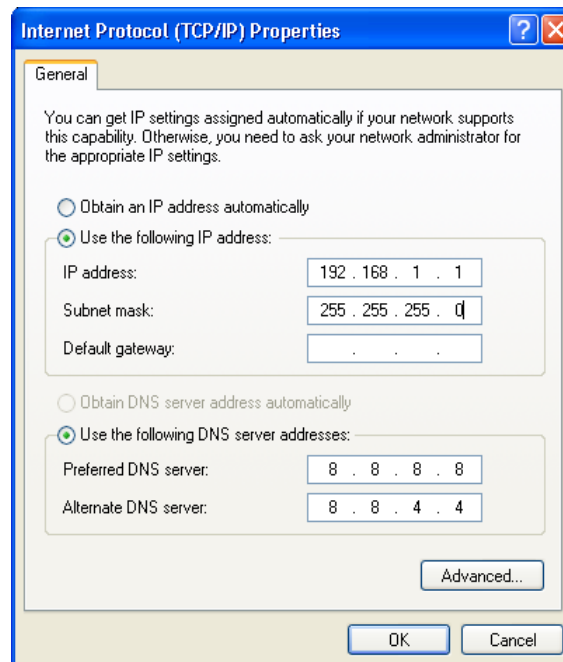
Hình 2.89: Bước 2 hướng dẫn sử dụng phần mềm tftpd32

- Một hộp thoại hiện ra, tại This connection uses the following items bạn chọn Internet Protocol (TCP/IP) và click vào Properties



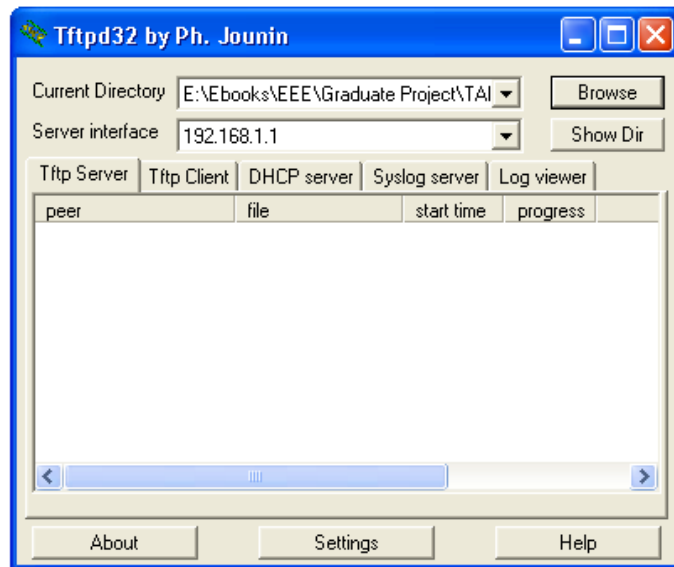
Hình 2.90: Bước 3 hướng dẫn sử dụng phần mềm tftpd32

- Một hộp thoại mới hiện ra, bạn chọn Use the following IP address và thiết lập IP tĩnh cho máy tính của bạn. Để dễ nhớ thì chúng ta dùng địa chỉ IP là 192.168.1.1 và Subnet mask là 255.255.255.0. Xong nhấn OK.



Hình 2.91: Bước 4 hướng dẫn sử dụng phần mềm tftpd32

- Tiếp theo ta kết nối KIT KM9260 với máy tính thông qua cổng Ethernet và mở chương trình tftpd32, một hộp thoại sẽ hiện ra. Tại Current Directory ta click vào Browse để chọn thư mục chứa file cần chép vào KIT KM9260. Tại Server interface ta nhập vào địa chỉ IP tĩnh của máy tính.



Hình 2.92: Bước 5 hướng dẫn sử dụng phần mềm tftpd32

- Bây giờ ta khởi động KIT KM9260 và nhập dòng lệnh sau vào Terminal của Linux

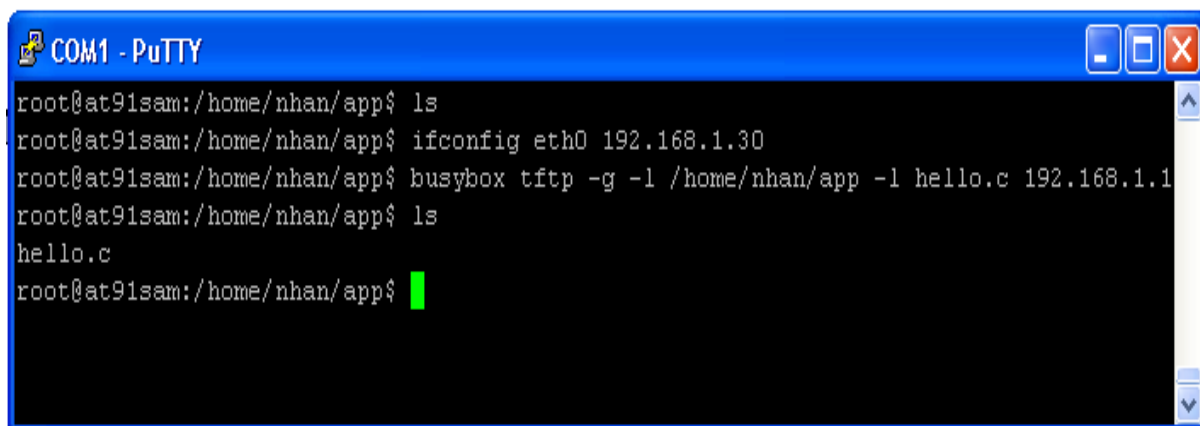
```
$ ifconfig eth0 192.168.1.30
```

```
$ busybox tftp -g -l /<thư mục lưu file ở trên kit> -l <tên file cần chép vào kit>
```

```
<địa chỉ IP tĩnh của máy tính>
```
- Lúc này file mà bạn muốn chép vào vùng nhớ của KIT KM9260 đã được thực hiện.

Ví dụ:

chúng ta chép file hello.c nằm trong thư mục đã được xác định ở trong phần Current Directory vào thư mục /home/nhan/app thì ta nhập lệnh như sau



```
COM1 - PuTTY
root@at91sam:/home/nhan/app$ ls
root@at91sam:/home/nhan/app$ ifconfig eth0 192.168.1.30
root@at91sam:/home/nhan/app$ busybox tftp -g -l /home/nhan/app -l hello.c 192.168.1.1
root@at91sam:/home/nhan/app$ ls
hello.c
root@at91sam:/home/nhan/app$
```

Hình 2.93: Ví dụ về sử dụng tftpd32 để chép một tập tin lên vùng nhớ của kit

E- Chương trình SSH SECURE SHELL CLIENT:

I. Giới thiệu:

Nếu chúng ta sử dụng phần mềm **VMware Workstation** với hệ điều hành trên máy ảo là Linux và hệ điều hành chính là Window thì chúng ta không thể chép dữ liệu trực tiếp từ Linux sang Window và từ Window sang Linux được. Để chép được dữ liệu qua lại giữa hai hệ điều hành này thì chúng ta phải dùng phần mềm SSH.

Phần mềm SSH sẽ kết nối hai hệ điều hành Window trên máy thật và Linux trên máy ảo thông qua hệ thống mạng ảo.

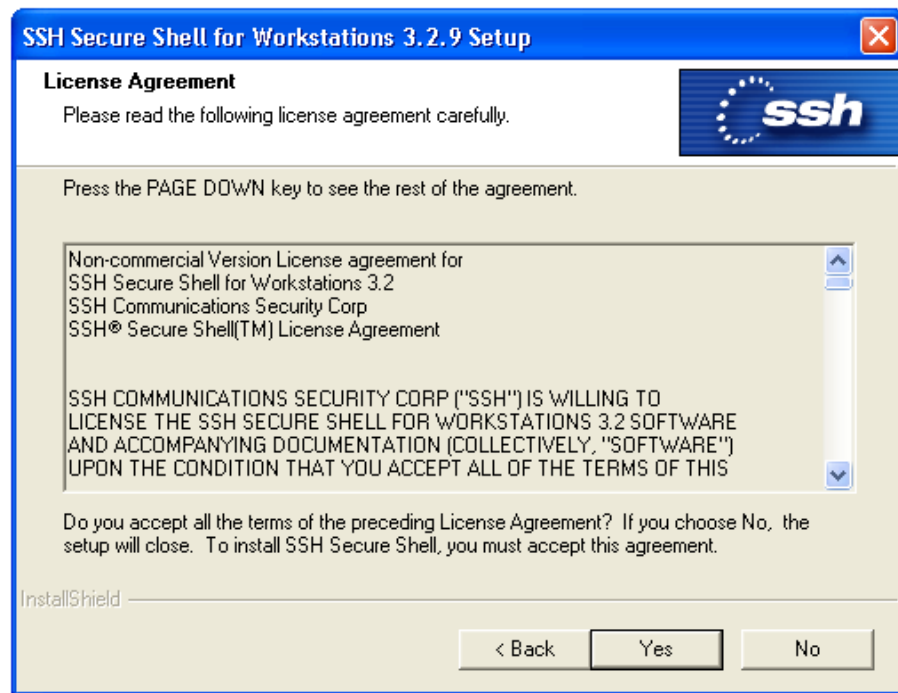
II. Hướng dẫn cài đặt SSH SECURE SHELL:

- Đầu tiên ta chạy file **SSHSecureShellClient-3.2.9.exe**, một hộp thoại hiện ra, ta nhấn Next để tiếp tục.



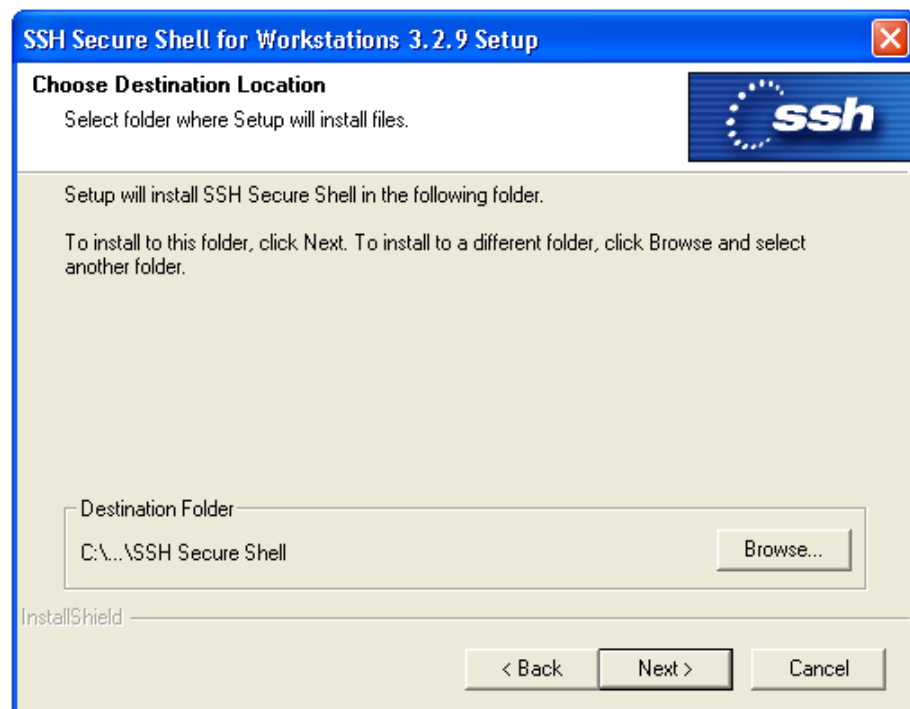
Hình 2.94: Bước 1 cài đặt SSH Secure Shell

- Hiện ra hộp thoại ta nhấn Yes



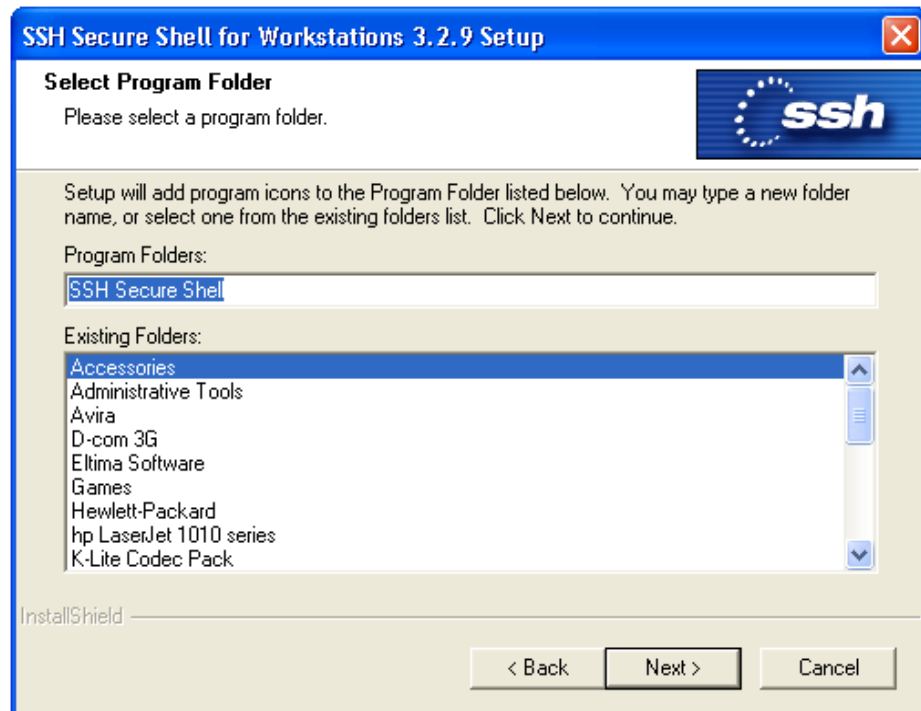
Hình 2.95: Bước 2 cài đặt SSH Secure Shell

- Hộp thoại tiếp theo hiện ra yêu cầu ta chọn nơi cài phần mềm. Chúng ta Click vào Browse và chọn vị trí cần cài đặt phần mềm. Vị trí mặc định là C:\Program Files\SSH Communications Security\SSH Secure Shell. Xong ta nhấn Next



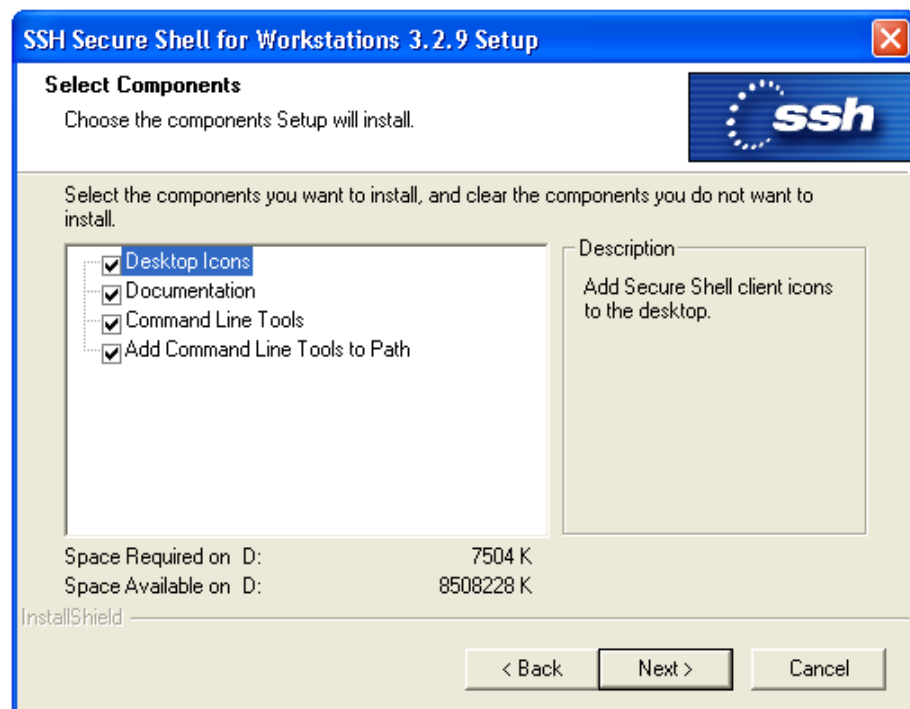
Hình 2.96: Bước 3 cài đặt SSH Secure Shell

- Hộp thoại tiếp theo hiện ra ta nhấn Next để tiếp tục.



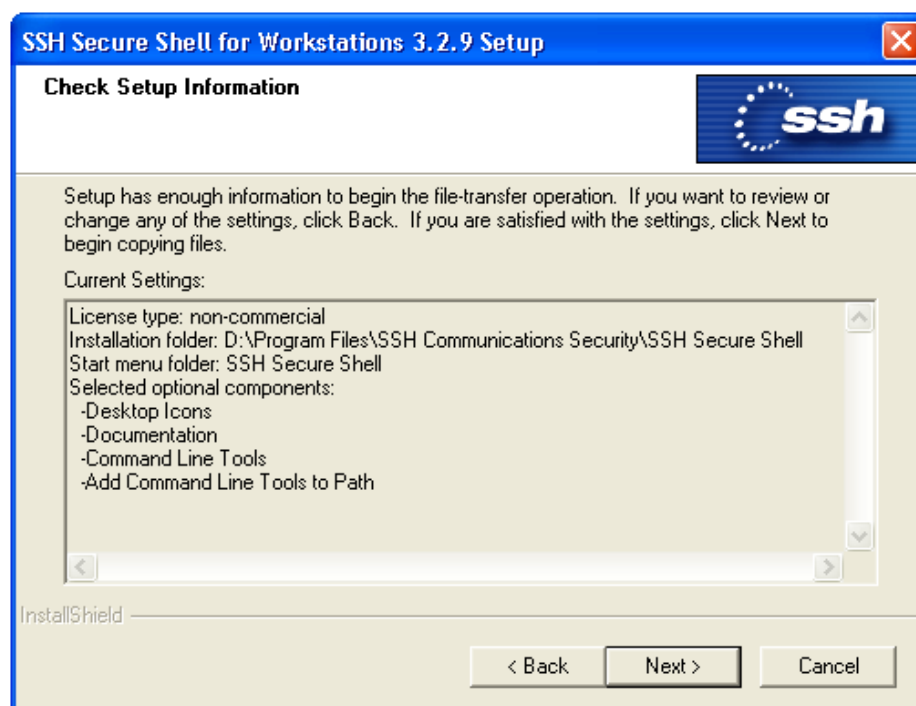
Hình 2.97: Bước 4 cài đặt SSH Secure Shell

- Ta tiếp tục nhấn Next.



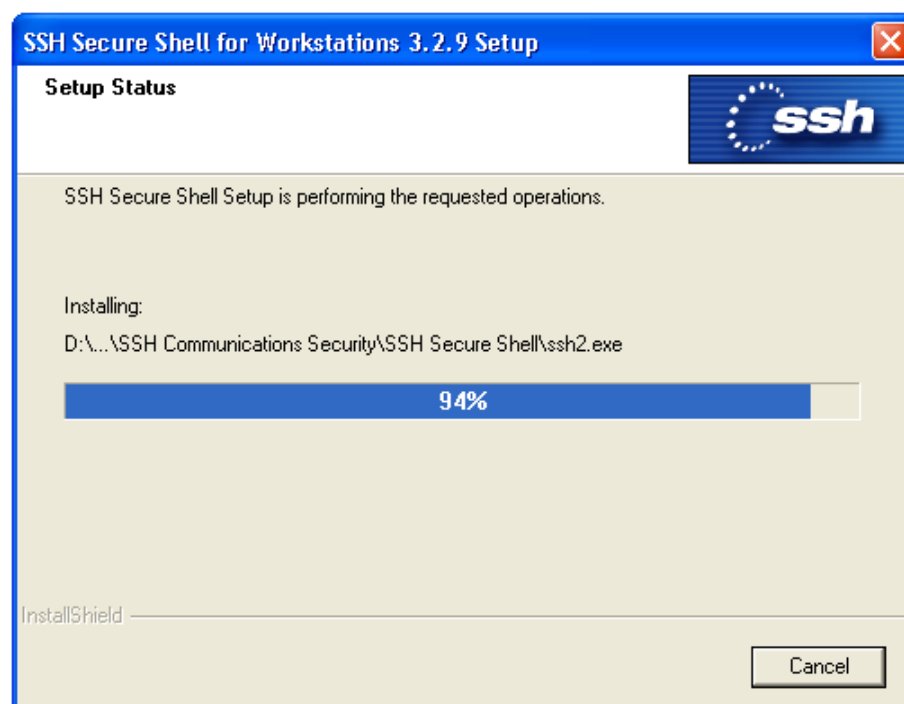
Hình 2.98: Bước 5 cài đặt SSH Secure Shell

- Nhấn Next



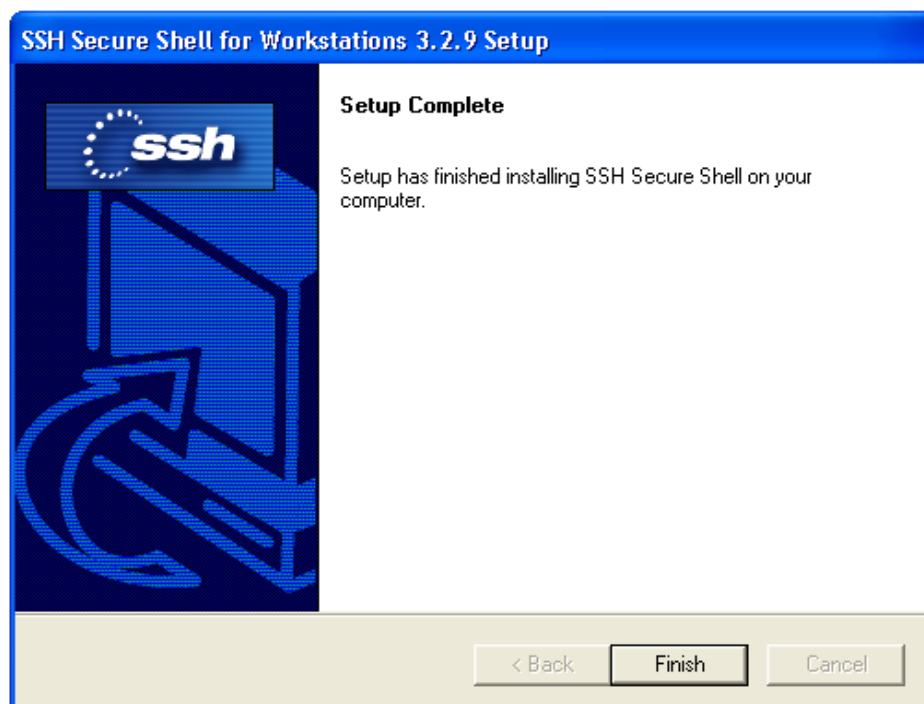
Hình 2.99: Bước 6 cài đặt SSH Secure Shell

- Hộp thoại hiện ra thông báo SSH đang được cài đặt.



Hình 2.100: Bước 7 cài đặt SSH Secure Shell

- Hộp thoại cuối cùng hiện ra thông báo SSH đã cài đặt xong. Ta nhấn Finish để hoàn thành quá trình cài đặt.



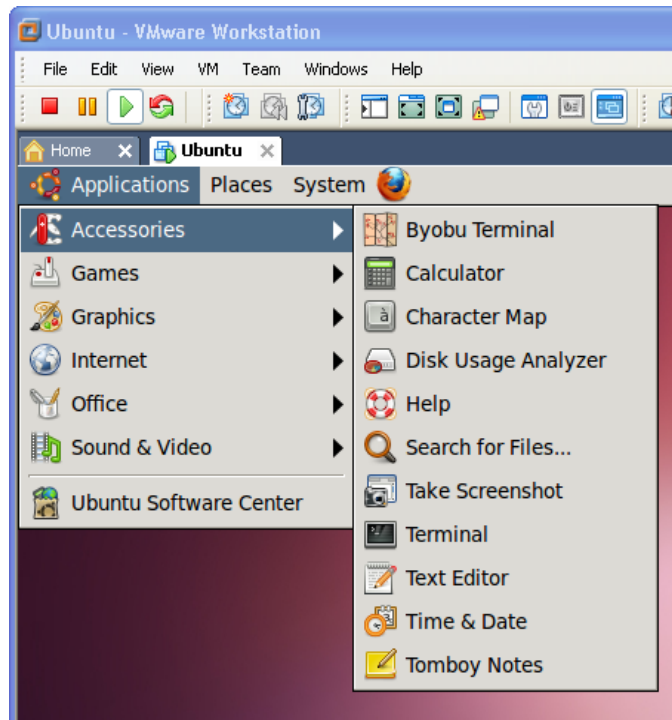
Hình 2.101: Bước 8 cài đặt SSH Secure Shell

- Nếu quá trình cài đặt thành công thì trên Desktop sẽ có Icon của chương trình SSH Shell Client



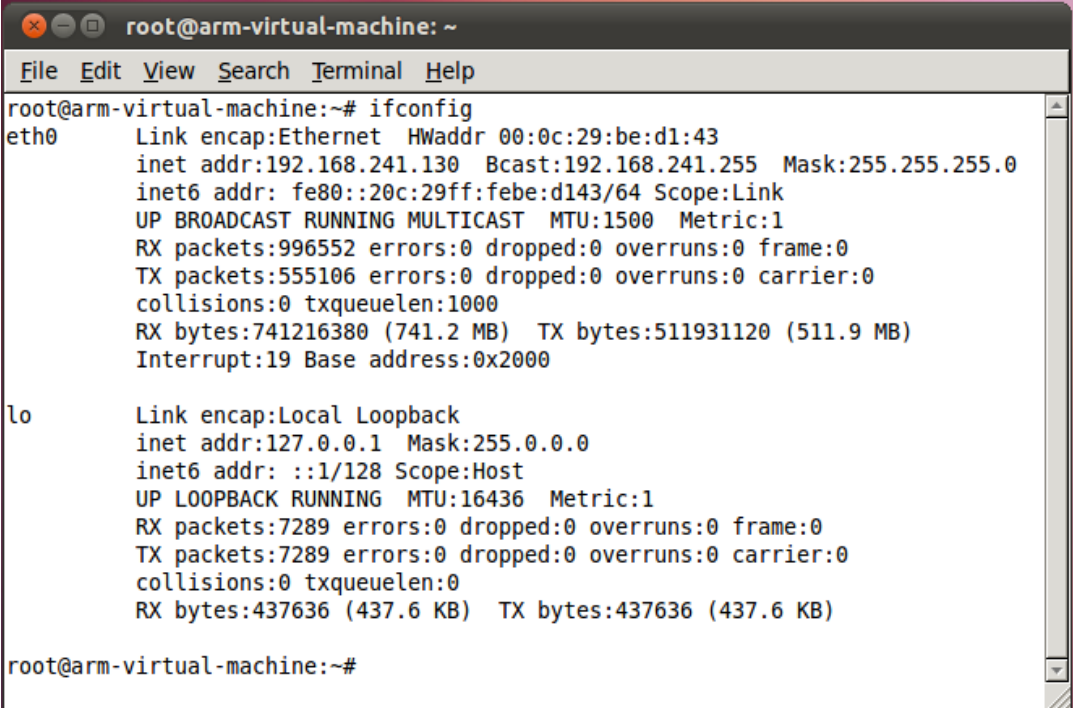
III. Hướng dẫn sử dụng SSH SECURE SHELL:

- Trước tiên thì ta mở phần mềm **VMware Workstation** và khởi động máy ảo. Nếu Linux hỏi User Name và Password thì bạn nhập Username và Password của bạn đã đặt từ trước. Ở đây chúng ta dùng User Name là **root** và Pass là **root00**. Trong máy vi tính ảo bạn chọn Applications/Accessories/Terminal để mở màn hình Terminal của Linux.



Hình 2.102: Bước 1 hướng dẫn sử dụng SSH Secure Shell

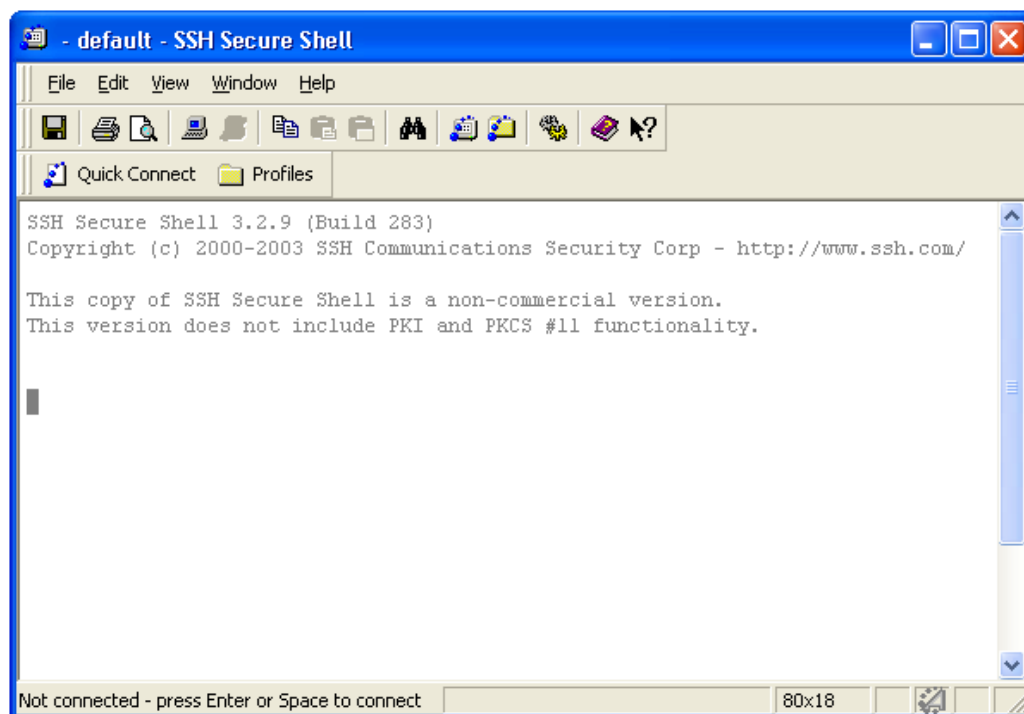
- Tại màn hình Terminal bạn nhập vào lệnh `ifconfig` để biết được địa chỉ IP của máy vi tính ảo. Theo hình sau thì địa chỉ IP của máy vi tính ảo là 192.168.241.130



```
root@arm-virtual-machine: ~  
File Edit View Search Terminal Help  
root@arm-virtual-machine:~# ifconfig  
eth0      Link encap:Ethernet  HWaddr 00:0c:29:be:d1:43  
          inet addr:192.168.241.130  Bcast:192.168.241.255  Mask:255.255.255.0  
          inet6 addr: fe80::20c:29ff:febe:d143/64 Scope:Link  
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
          RX packets:996552 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:555106 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:741216380 (741.2 MB)  TX bytes:511931120 (511.9 MB)  
          Interrupt:19 Base address:0x2000  
  
lo        Link encap:Local Loopback  
          inet addr:127.0.0.1  Mask:255.0.0.0  
          inet6 addr: ::1/128 Scope:Host  
          UP LOOPBACK RUNNING  MTU:16436  Metric:1  
          RX packets:7289 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:7289 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:0  
          RX bytes:437636 (437.6 KB)  TX bytes:437636 (437.6 KB)  
  
root@arm-virtual-machine:~#
```

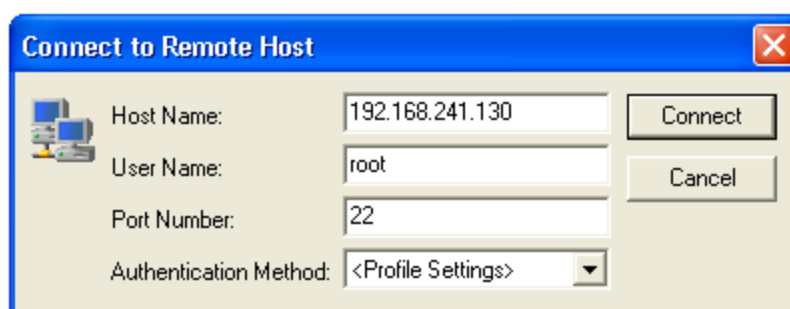
Hình 2.103: Bước 2 hướng dẫn sử dụng SSH Secure Shell

- Tiếp theo bạn chạy chương trình SSH Secure Shell. Giao diện chính của chương trình hiện ra.



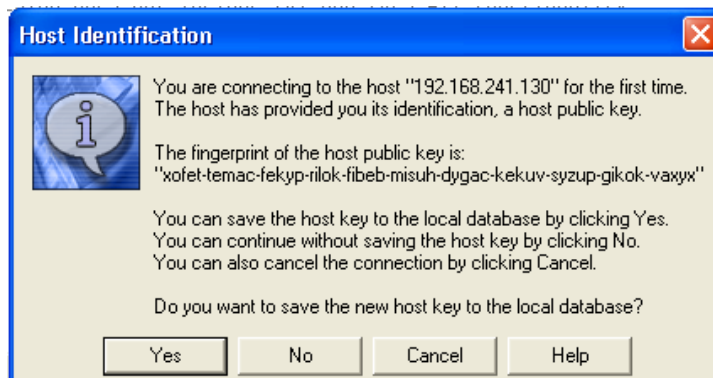
Hình 2.104: Bước 3 hướng dẫn sử dụng SSH Secure Shell

- Click vào Quick Connect, một hộp thoại sẽ hiện ra. Tại Host Name bạn nhập địa chỉ IP của máy vi tính ảo. Tại User Name bạn nhập User Name của máy vi tính ảo. Xong Click **Connect**



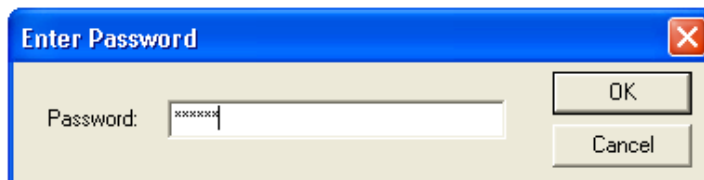
Hình 2.105: Bước 4 hướng dẫn sử dụng SSH Secure Shell

- Một hộp thoại hiện ra hỏi chúng ta có muốn lưu địa chỉ IP của máy vì tính ảo lại không. Để khỏi mất thời gian cho các thao tác sau này chúng ta nên chọn **Yes**



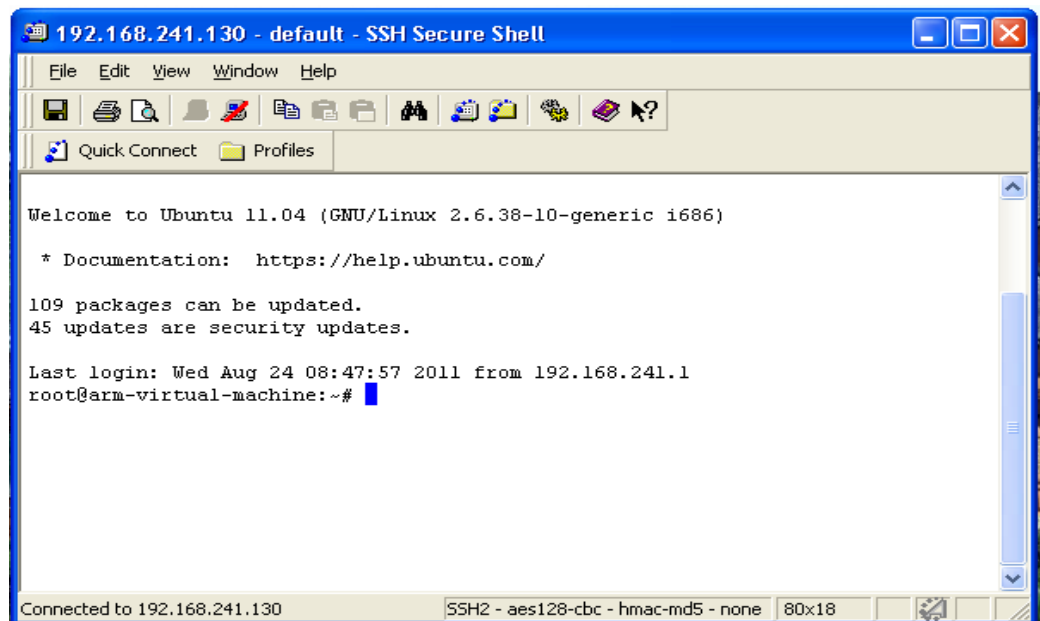
Hình 2.106: Bước 5 hướng dẫn sử dụng SSH Secure Shell

- Hộp thoại tiếp theo hiện ra yêu cầu ta nhập Password của hệ điều hành trên máy ảo. Ở đây ta nhập **root00**. Xong ta nhấn **OK**




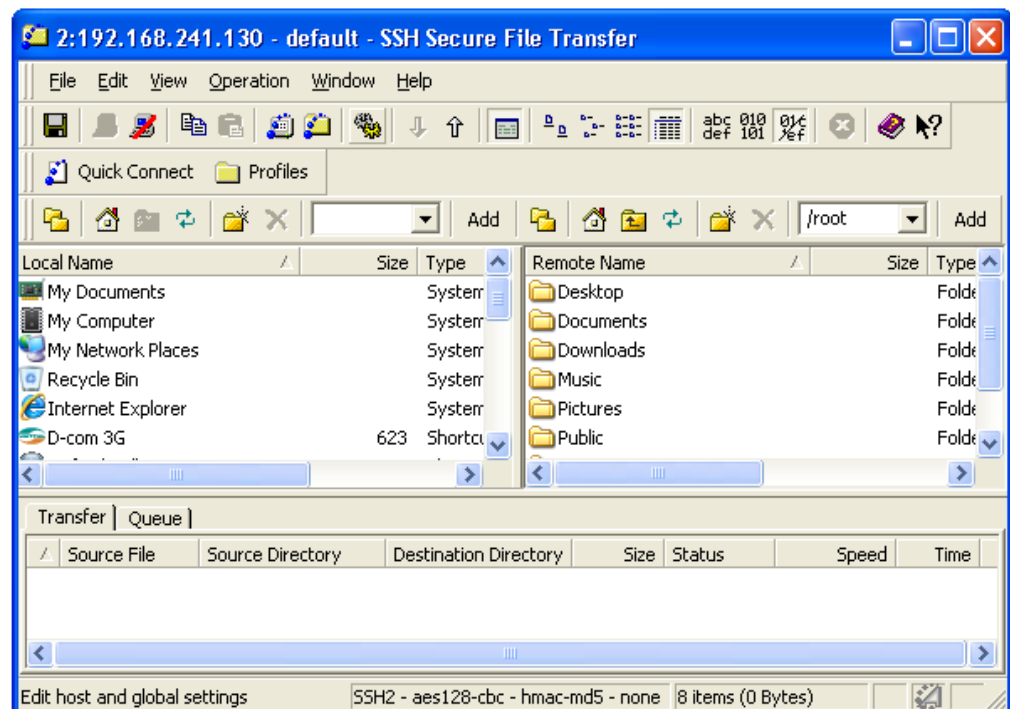
Hình 2.107: Bước 6 hướng dẫn sử dụng SSH Secure Shell

- Nếu kết nối thành công thì trên giao diện chính của SSH Secure Shell sẽ hiện ra màn hình Terminal của Linux ở máy ảo. Màn hình Terminal này có chức năng giống hoàn toàn màn hình Terminal của Linux ảo, nên chúng ta có thể thực thi các dòng lệnh từ Terminal này để biên dịch chương trình bằng Linux.




Hình 2.108: Bước 7 hướng dẫn sử dụng SSH Secure Shell

- Từ giao diện chính của SSH Secure Shell bạn click vào biểu tượng  (New File Transfer Window), một cửa sổ mới hiện ra



Hình 2.109: Bước 8 hướng dẫn sử dụng SSH Secure Shell

- Chúng ta có thể thấy cửa sổ mới hiện ra được chia làm 2 phần. Phần bên trái chính là các file và thư mục của hệ điều hành Window trên máy vi tính thật. Còn phần bên phải là các file và thư mục của hệ điều hành Linux ảo. Chúng ta có thể chọn các file và thư mục bên Linux ảo và kéo thả sang Window và ngược lại. Hoặc click chuột phải vào file bên Linux và chọn Copy, sau đó click bên Window và click vào biểu tượng  Past.

C- Kết luận:

Các phần mềm nói trên tương đối dễ cài đặt và dễ sử dụng. Người học sau khi đọc phần này không cần phải học thuộc lòng từng bước, từng thao tác, mà chỉ cần nhớ các bước này để phục vụ cho quá trình thao tác với kit nhúng.

Mỗi phần mềm đều có một chức năng, nhiệm vụ riêng. Vì vậy để sử dụng được kit nhúng KM9260 thì chúng ta không thể thiếu một trong các phần mềm trên. Người học cần phải đảm bảo là các phần mềm đã được cài đặt thành công và hoạt động tốt, nếu gặp bất kỳ sự cố nào trong quá trình thao tác thì có thể nhờ sự hỗ trợ của giáo viên.

Như vậy, sau khi đọc phần này người học đã phần nào nắm bắt được chức năng nhiệm vụ của các phần mềm. Tuy nhiên việc ứng dụng cụ thể các phần mềm này như thế nào thì chúng ta sẽ tìm hiểu trong phần tiếp theo của giáo trình.

BÀI 5

THAO TÁC TRÊN PHẦN MỀM HỆ THỐNG NHÚNG

Trong phần này chúng ta sẽ tìm hiểu cụ thể việc ứng dụng các phần mềm đã trình bày trong các phần trước như thế nào.

Một trong những thao tác quan trọng để kit nhúng có thể hoạt động được đó là nạp các phần mềm hệ thống vào vùng nhớ tương ứng trên kit và cài đặt các thông số ban đầu cho kit. Trong phần này sẽ trình bày từng bước, từng thao tác để thực hiện được nhiệm vụ này.

Ngoài ra, trong phần này chúng ta sẽ tìm hiểu các cách biên dịch phần mềm, chạy phần mềm. Những thao tác này là thực sự cần thiết để chúng ta thực hiện các ứng dụng sau này.

I. Trình biên dịch chéo Cross Toolchians:

Trình biên dịch này chạy trên hệ điều hành Linux. Là trình biên dịch cho AT91SAM. Cũng giống như các vi xử lý khác, AT91SAM không hiểu được ngôn ngữ C mà chúng ta dùng để viết các chương trình ứng dụng, vì vậy chúng ta cần phải chuyển chương trình mà chúng ta viết bằng ngôn ngữ C sang ngôn ngữ mà AT91SAM có thể hiểu được. Ở đây chúng ta dùng trình biên dịch chéo CROSS TOOLCHIANS.

Trình biên dịch chéo CROSS TOOLCHIANS được nhà sản xuất cung cấp với file nén dưới tên arm-2011.03-41-arm-none-linux-gnueabi-.tar.bz2

Để cài đặt và sử dụng trình biên dịch chéo CROSS TOOLCHIANS trên hệ điều hành Linux ta làm theo các bước sau:

- + Mở phần mềm VMWare Workstation và khởi động máy ảo có cài hệ điều hành Linux. Sau đó ta mở màn hình Terminal của Linux.
- + Từ màn hình Terminal ta tạo thư mục làm việc như sau:

```
/$ mkdir /home/nhan/compiler
```
- + Dùng phần mềm SSH Secure Shell để chép file arm-2011.03-41-arm-none-linux-gnueabi-.tar.bz2 vào thư mục /home/nhan/compiler vừa mới tạo.
- + Giải nén file arm-2011.03-41-arm-none-linux-gnueabi-.tar.bz2

```
/$ cd /home/nhan/compiler
```



```
/home/nhan/compiler$ tar -jxvf arm-2011.03-41-arm-none-linux-gnueabi-.tar.bz2
```

****Chú ý:** Câu lệnh này được nằm trên cùng một dòng.

- + Sau khi giải nén xong chúng ta sẽ được thư mục /home/nhan/compiler/arm-2011.03. Đây là thư mục chứa trình biên dịch chéo.
- + Để có thể sử dụng được trình biên dịch chéo này thì ta phải cài đặt lại biến môi trường của Linux

```
/home/nhan/compiler$ export PATH = $PATH:  
/home/nhan/compiler/arm-2011.03/bin
```

****Chú ý:** Câu lệnh này được nằm trên cùng một dòng.

- + Để ý rằng trong thư mục /home/nhan/compiler/arm-2011.03/bin chứa những file cần thiết phục vụ cho việc biên dịch cho KM9260. Các file này có tên đều được bắt đầu bằng chuỗi “arm-none-linux-gnueabi-“ và phần còn lại của tên file chính là các lệnh trong Linux thường dùng để biên dịch như: gcc,... và tên của các file này cũng chính là các lệnh mà chúng ta sử dụng trong quá trình biên dịch.
- + Ta có thể kiểm tra việc export biến môi trường PATH thành công hay không bằng cách dùng lệnh như sau:

```
/$ arm-none-linux-gnueabi-gcc  
/$ arm-none-linux-gnueabi-gcc: no input file
```
- + Nếu có xuất hiện thông báo “no input files” thì thao tác export trên thành công, nếu có thông báo “command not found” có nghĩa là ta chưa export đúng đường dẫn. Chú ý rằng bước export PATH rất quan trọng, mỗi lần khởi động máy Linux hoặc mở một màn hình Terminal khác ta bắt buộc phải export biến môi trường lại để đảm bảo toolchains có thể hoạt động được.
- + Từ bây giờ khi nhắc đến biên dịch bất kỳ chương trình gì cho KIT KM9260 thì chúng ta đều dùng đến chương trình biên dịch chéo này. Các lệnh cụ thể của trình biên dịch này sẽ được trình bày trong các phần sau.

II. Các bước biên dịch Kernel:

Trước tiên chúng ta phải tìm hiểu xem tại sao lại phải biên dịch Kernel?

Các nhà sản xuất hoặc các công ty lập trình cung cấp cho chúng ta Kernel được viết theo ngôn ngữ C. Nên vi xử lý không thể đọc được các dữ liệu này. Vì vậy chúng ta phải biên dịch Kernel này thành một file ảnh mà vi xử lý có thể đọc được. Hơn nữa, trong Kernel có rất nhiều chức năng không được kích hoạt. Vì vậy khi muốn sử dụng chức năng nào đó chưa được kích hoạt thì chúng ta phải chỉnh sửa lại Kernel. Và khi đó chúng ta phải biên dịch lại Kernel thì hệ thống mới hoạt động được.

Các bước này được thực hiện trong máy ảo chứa hệ điều hành Linux. Các bước biên dịch kernel được tiến hành như sau:

- Để bắt đầu quá trình biên dịch Kernel thì ta tiến hành cài đặt chương trình biên dịch chéo CROSS TOOLCHAINS như hướng dẫn ở trên.
- Chúng ta chuẩn bị các file như sau:
 - + File linux-2.6.27.tar.bz2 chép vào thư mục /home/nhan/kernel
 - + File linux-2.6.27-km9260-11102009.diff chép vào thư mục /home/nhan/kernel
 - + File uboot-mkimage.zip chép vào thư mục /home/nhan/kernel
- Cài chương trình mkimage là chương trình dùng để tạo ra uImage. uImage chính là file ảnh của Kernel Linux. Các bước cài đặt mkimage được thực hiện như sau:
 - + Giải nén file uboot-mkimage.zip

```
/home/nhan/kernel $ tar -zxvf uboot-mkimage.zip
```
 - + Vào thư mục uboot-mkimage mới vừa giải nén tiến hành biên dịch file mkimage

```
/home/nhan/kernel $ cd uboot-mkimage
/home/nhan/kernel/uboot-mkimage $ make clean
/home/nhan/kernel/uboot-mkimage $ make
```
 - + Cài đặt quyền truy xuất file mkimage cho tất cả các nhóm

```
/home/nhan/kernel/uboot-mkimage $ chmod 777 mkimage
```
 - + Chép file mkimage vào thư mục /bin của hệ điều hành

```
/home/nhan/kernel/uboot-mkimage $ cp mkimage /bin
```
- Giải nén source của Kernel

```
/home/nhan/kernel $ tar -jxvf kernel-2.6.27.tar.bz2
```

- Vào thư mục vừa giải nén và tiến hành patch file linux-2.6.27-km9260-11102009.diff vào thư mục này.

```
/home/nhan/kernel $ cd linux-2.6.27
```

```
/home/nhan/kernel/linux-2.6.27 $ cat ../linux-2.6.27-km9260-11102009.diff |  
patch -p1
```

- Tiếp theo ta cấu hình Kernel

```
/home/nhan/kernel/linux-2.6.27$ make ARCH=arm CROSS_COMPILE=arm-  
none-linux-gnueabi- km9260_defconfig
```

- Biên dịch Kernel

```
/home/nhan/kernel/linux-2.6.27$ make ARCH=arm CROSS_COMPILE=arm-  
none-linux-gnueabi- uImage
```

- Sau khi biên dịch xong thì ta được file uImage trong thư mục /home/nhan/kernel/linux-2.6.27/arch/arm/boot. Đây chính là file ảnh của Kernel mà ta dùng để nạp vào KIT KM9260.

****Lưu ý:** Các lệnh trên được lập trình trên cùng một dòng lệnh shell.

III. Chỉnh sửa Kernel:

Chỉnh sửa Kernel thực chất là chúng ta thêm hoặc bớt các chức năng của Kernel. Vì trong Kernel tích hợp rất nhiều chức năng, nên để giảm kích thước cho Kernel và thuận tiện trong quá trình quản lý thì các chức năng của Kernel sẽ không được kích hoạt toàn bộ. Mà chỉ một số chức năng thông dụng mới được kích hoạt. Còn các chức năng mà trong ứng dụng cần đến thì chúng ta phải thêm các chức năng ấy vào trong Kernel (cụ thể từng chức năng của Kernel như thế nào thì trong các phần sau sẽ trình bày).

Việc chỉnh sửa Kernel được thực hiện trong máy Linux ảo.

Để thuận tiện cho việc trình bày thì chúng ta mặc định trình biên dịch chéo CROSS TOOLCHAINS đã được cài đặt như trên, thư mục /home/nhan/kernel/linux-2.6.27 đã tồn tại.

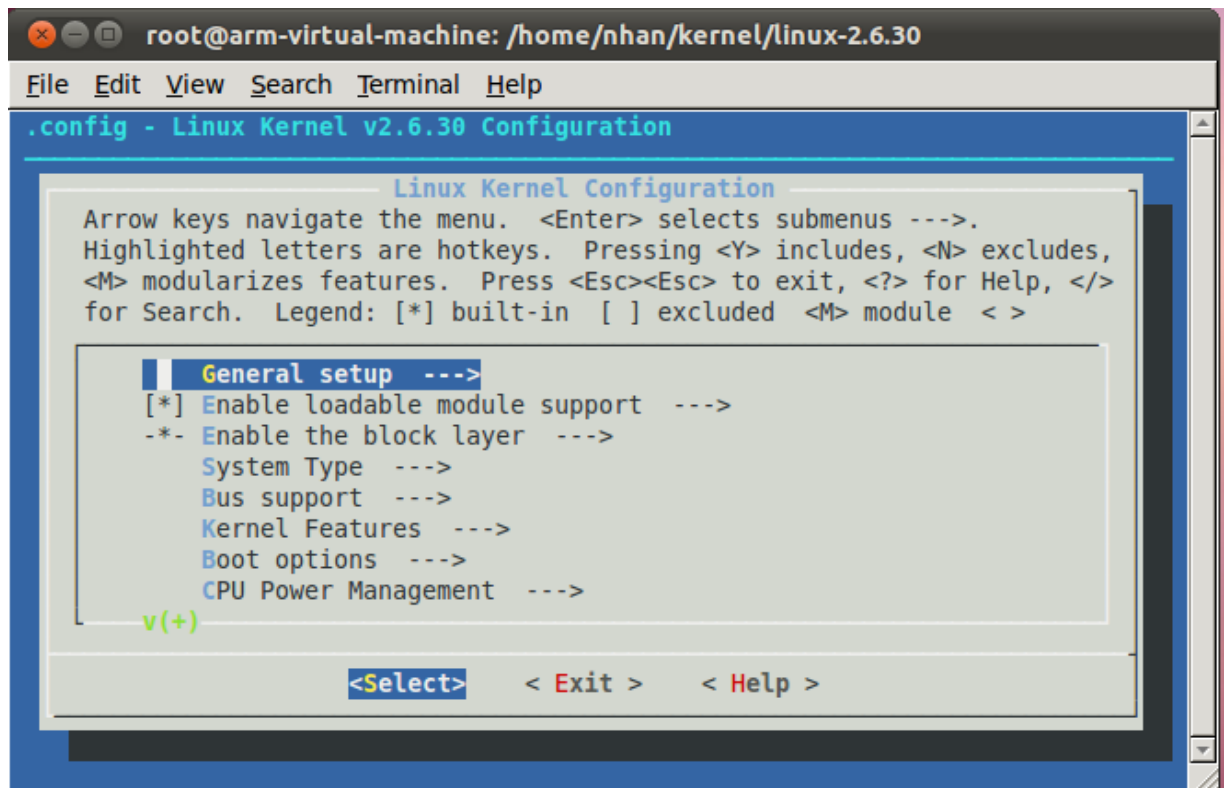
Để chỉnh sửa các thông tin của kernel, ta tiến hành theo các bước:

- Chúng ta cài đặt biến môi trường như trình bày ở phần trên.
- Trong thư mục /home/nhan/kernel/linux-2.6.27 chúng ta nhập dòng lệnh:

```
/home/nhan/kernel/linux-2.6.27$ make ARCH=arm  
CROSS_COMPILER=arm-none-linux-gnueabi- menuconfig
```

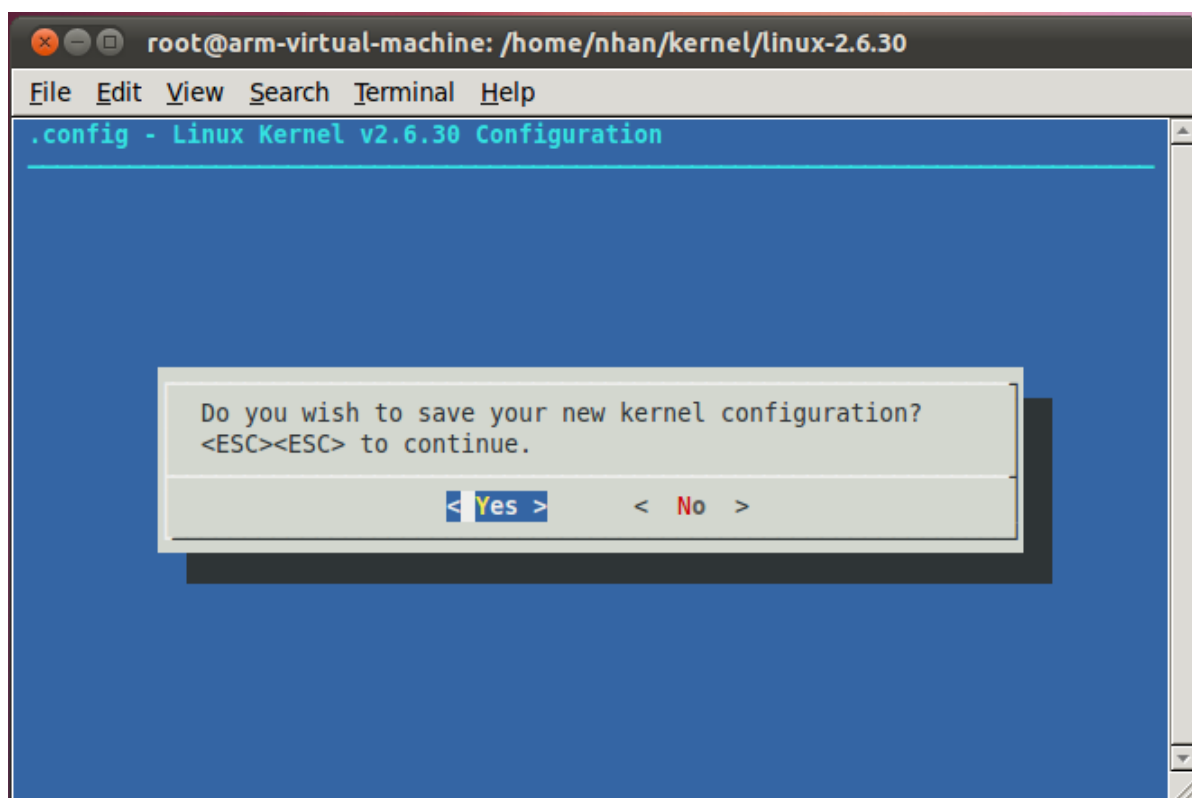
***Lệnh được nhập trên cùng một dòng.*

- Lúc đó giao diện của chương trình Linux Kernel Configuration hiện ra. Chúng ta chỉnh sửa Kernel thông qua giao diện này.



Hình 2.110: Giao diện chương trình Linux Kernel Configuration

- Trong Linux Kernel Configuration chúng ta dùng phím mũi tên lên xuống của bàn phím để di chuyển thanh sáng lên xuống chọn các mục. Các phím mũi tên trái phải dùng để chọn các lệnh <select> <exit> <help> ở hàng phía dưới cùng của giao diện. Nhấn Enter để thực thi lệnh hoặc truy cập.
- Nếu trước các mục mà có dấu [] hoặc < > thì mục đó chúng ta có thể chọn kích hoạt hoặc không kích hoạt chức năng tương ứng của mục bằng cách nhấn phím Space trên bàn phím máy tính. Lúc đó chúng ta sẽ thấy phần đầu của mục thay đổi. Nếu là dấu [] hoặc < > thì là chưa kích hoạt, [*] hoặc <*> là kích hoạt.
- Nếu trước các mục mà không có dấu [] hoặc < > thì mục đó sẽ có các mục ở bên trong nữa (giống với thư mục). Để truy cập vào bên trong mục thì ta thực thi lệnh <Select>, để xuất ra khỏi thư mục ta thực thi lệnh <Exit>. Tại mục giống như hình trên, có sự thay đổi, chúng ta dùng lệnh <Exit> thì chương trình sẽ hỏi chúng ta có muốn lưu sự thay đổi hay không. Chúng ta chọn Yes nếu muốn lưu lại và chọn No nếu không muốn lưu lại.



Hình 2.111: Thông báo khi thoát khỏi chương trình Linux Kernel Configuration

- Như vậy, ta có thể thấy trong Linux Kernel Configuration có rất nhiều mục tương ứng với từng chức năng để chúng ta chọn kích hoạt hoặc không kích hoạt.

Ngoài ra còn rất nhiều chức năng khác mà chúng ta sẽ tìm hiểu trong quá trình thực hiện các ứng dụng cụ thể ở các phần sau.

IV. Các biến môi trường và lệnh cơ bản của U-boot:

Như chúng ta đã biết U-boot có nhiệm vụ tìm Kernel, chép Kernel vào trong SDRAM và giao quyền thực thi lại cho Kernel. Trong U-boot có hỗ trợ một số lệnh cơ bản và các biến môi trường để phục vụ cho việc thực thi nhiệm vụ của U-boot.

Các lệnh cơ bản của u-boot:

- **tftp diachi tenfile:** diachi là địa chỉ vật lý của vùng nhớ SDRAM cần ghi dữ liệu, tenfile là tên file cần nạp xuống. File cần nạp xuống nằm trong thư mục mà chương trình tftpd32 đã chọn. Như vậy trong u-boot lệnh tftp dùng để nạp một file trực tiếp vào SDRAM của KIT.
- **go diachi:** có chức năng chạy một chương trình ứng dụng standalone tại diachi là địa chỉ vật lý vùng nhớ SDRAM.

- ***bootm diachi*** : có chức năng chạy Kernel tại diachi là địa chỉ vật lý của SDRAM.
 - ***nand erase diachi dungluong*** : có chức năng xóa bộ nhớ NAND Flash từ diachi là địa chỉ offset của NAND Flash với dungluong Byte là dung lượng cần xóa.
 - ***nand read diachi1 diachi2 dungluong***: có chức năng đọc dungluong Byte tại địa chỉ offset diachi2 của NAND Flash lên địa chỉ vật lý diachi1 của SDRAM.
 - ***nand write diachi1 diachi2 dungluong***: có chức năng ghi dungluong Byte vào địa chỉ offset diachi2 của NAND Flash từ địa chỉ vật lý diachi1 của SDRAM
 - ***reset***: có chức năng reset hệ thống. Lệnh này tương đương với thao tác nhấn nút reset trên KIT.
 - ***print***: dùng để hiển thị các biến môi trường trên U-boot.
 - ***setenv tenbien giatri***: có chức năng cài đặt giatri cho biến môi trường tenbien.
 - ***save***: dùng để lưu các biến môi trường trên U-boot.
 - ***usb reset***: có chức năng quét cổng usb.
 - ***usb stop***: có chức năng remove usb.
- Các biến môi trường thông dụng trong U-boot:
- ***ipaddr***: biến này chứa ip của KIT
 - ***serverip***: biến này chứa ip của server mà KIT kết nối tới.
 - ***netmask***: chứa subnet mask của KIT.
 - ***bootdelay***: chứa thời gian delay với đơn vị là giây cho quá trình chạy tự động của U-boot.
 - ***baudrate***: dùng để setup tốc độ baud cho AURT
 - ***bootcmd***: biến chứa các lệnh chạy tự động khi khởi động.
 - ***bootargs***: biến này chứa các tham số trong quá trình kết gán rootfs.
 - ***root***: biến chứa đường dẫn của rootfs.
 - ***console***: biến chứa các thông số của cổng giao tiếp màn hình Terminal

V. Cài đặt cho hệ thống:

Để KIT KM9260 có thể hoạt động thì chúng ta phải cài đặt bootstrapcode, U-boot, Kernel và Rootfs vào các vùng nhớ tương ứng trên KIT KM9260 theo phân vùng bộ nhớ thích hợp.

Phần này chúng ta sẽ tìm hiểu các bước cài đặt trên KIT KM9260 để KIT có thể hoạt động.

KIT KM9260 chúng ta dùng có phân vùng bộ nhớ loại 1.

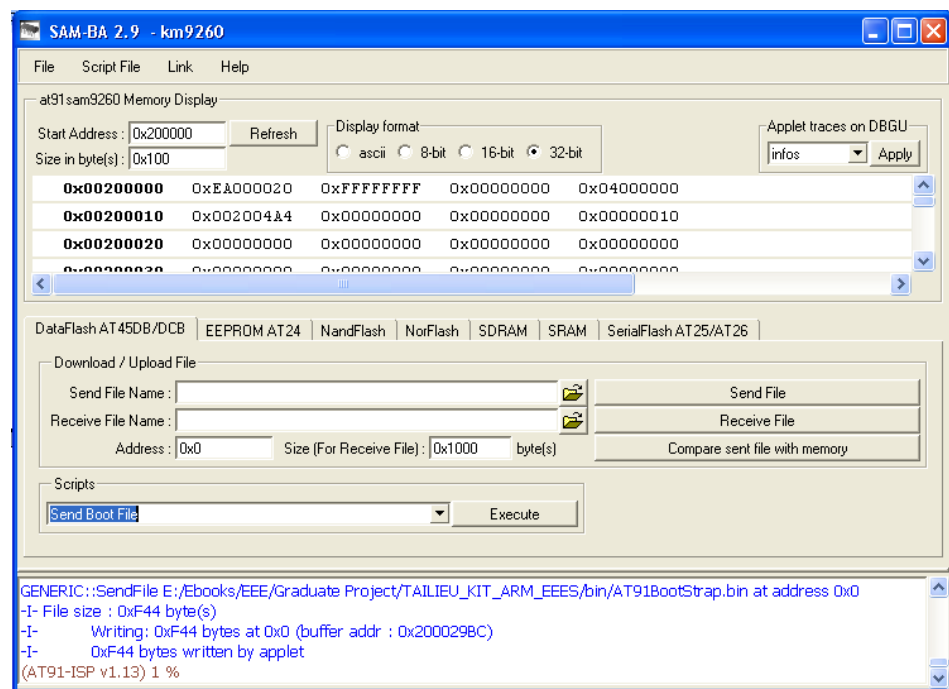
1. Cài đặt BootstrapCode:

File bootstrapcode để cài đặt cho hệ thống là file AT91BootStrap.bin.

Các bước thực hiện như sau:

- Kết nối KIT KM9260 với máy tính qua cổng USB Device.
- Rút Jump J15 trên KIT ra. Sau đó ta cấp nguồn khởi động cho KIT. Lúc này KIT sẽ chuyển sang chế độ SAMBA Boot và trên máy tính sẽ nhận diện KIT như là một thiết bị ngoại vi.
- Sau khi máy tính đã nhận ra KIT thì ta gắn Jump J15 lại để có thể chép dữ liệu vào Data Flash.
- Chúng ta mở chương trình SAMBA trên máy tính và nạp file AT91BootStrap.bin vào Data Flash với Address là 0x0.

Nếu nạp thành công thì giao diện chính của SAMBA như sau:



Hình 2.112: Giao diện chính của SAMBA khi nạp thành công BootstrapCode lên vùng nhớ của kit KM9260

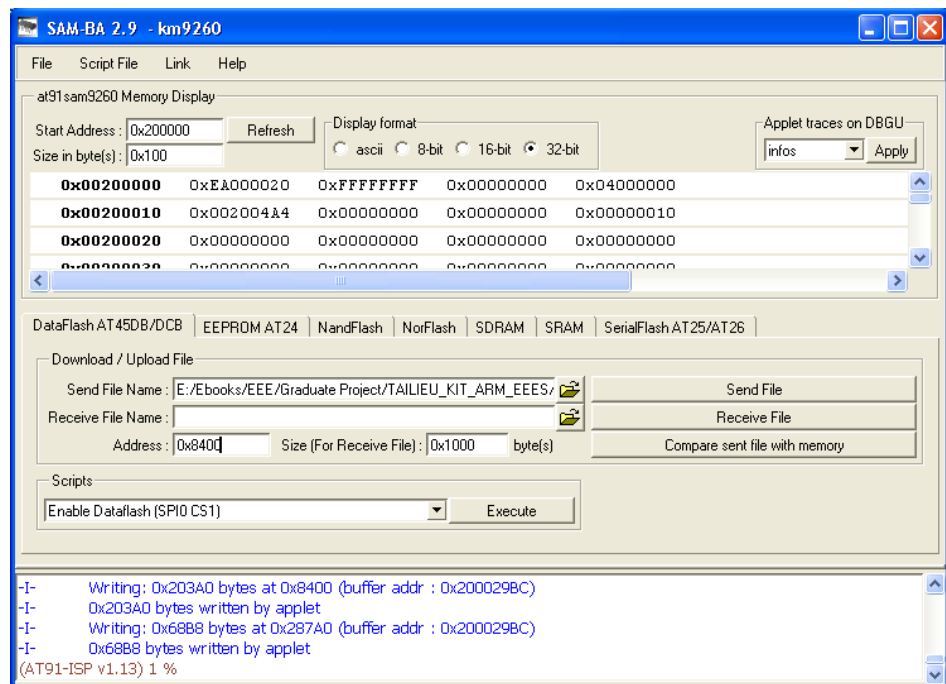
2. Cài đặt U-boot:

Sau khi nạp thành công bootstrapcode thì chúng ta tiến hành nạp U-boot.

File U-boot dùng để nạp vào KIT có tên u-boot.bin.

Từ chương trình SAMBA chúng ta nạp U-boot vào Data Flash với Address là 0x8400.

Nếu nạp thành công thì giao diện chính của SAMBA như sau:



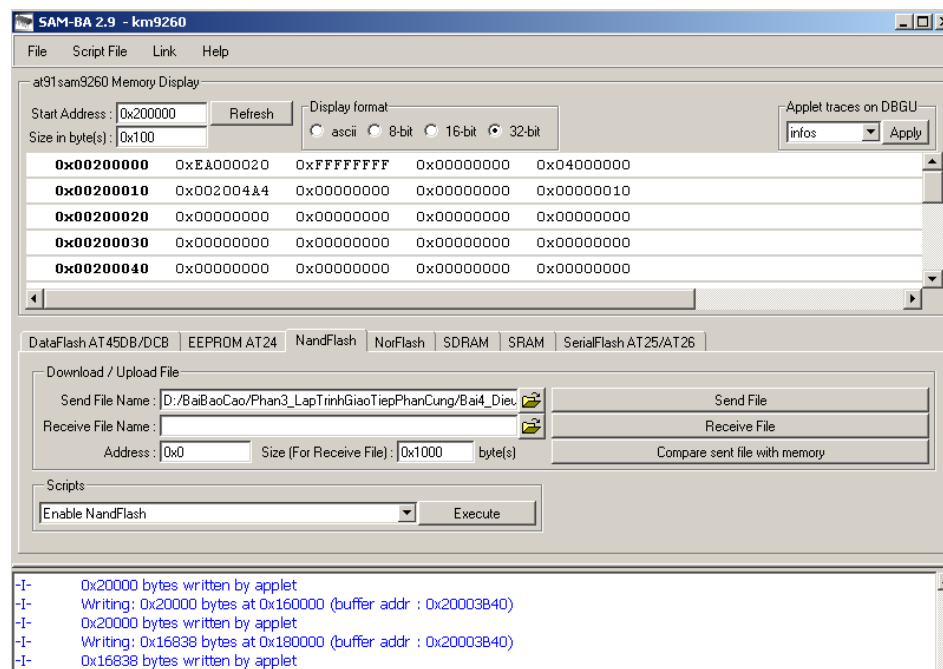
Hình 2.113: Giao diện chính của SAMBA khi nạp thành công U-boot lên vùng nhớ của kit KM9260

3. Cài đặt kernel:

Tiếp theo ta tiến hành cài đặt Kernel. File ảnh của Kernel dùng để nạp vào KIT là uImage mà chúng ta đã biên dịch ở phần trước.

Từ chương trình SAMBA chúng ta nạp file ảnh uImage vào NAND Flash với Address là 0x0.

Nếu chúng ta nạp thành công thì sẽ được giao diện sau:



Hình 2.114: Giao diện chính của SAMBA khi nạp thành công Kernel lên vùng nhớ của kit KM9260

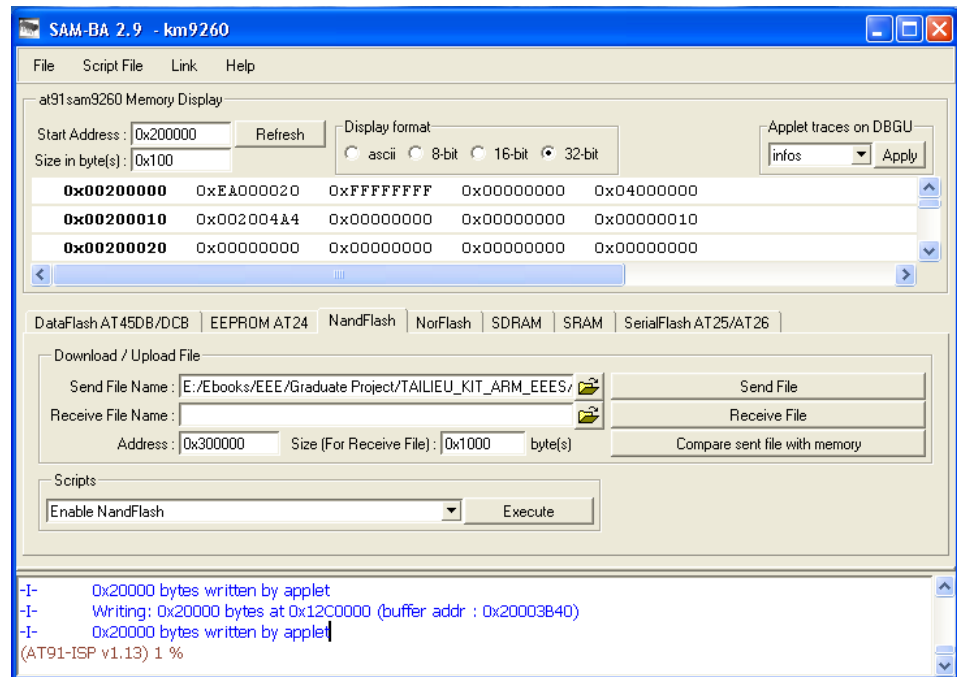
4. Cài đặt Rootfs:

Có 2 cách cài đặt cho rootfs là nạp rootfs vào NAND Flash và nạp vào thẻ Micro SD. Tùy vào dung lượng của rootfs và Kernel mà chúng ta lựa chọn một trong hai cách nạp rootfs. Nếu rootfs và Kernel có tổng dung lượng nhỏ hơn dung lượng của NAND Flash thì chúng ta có thể nạp rootfs vào NAND Flash. Còn nếu tổng dung lượng của rootfs và Kernel mà lớn hơn dung lượng của NAND Flash thì chúng ta nạp rootfs vào thẻ Micro SD.

- Cách nạp rootfs vào NAND Flash:

- + File rootfs dùng để nạp vào NAND Flash có tên Angstrom-console-image-demo-glibc.rootfs.jffs2.

- + Từ chương trình SAMBA chúng ta nạp file Angstrom-console-image-demo-glibc.rootfs.jffs2 vào NAND Flash với Address là 0x300000.
- + Nếu nạp thành công thì SAMBA sẽ thông báo như sau:



Hình 2.115: Giao diện chính của SAMBA khi nạp thành công Rootfs lên vùng nhớ của kit KM9260

• Nạp rootfs vào thẻ Micro SD:

- + Rootfs dùng để nạp vào Micro SD có tên là Angstrom-km9260-01252010.tar.bz2.
- + Ta kết nối thẻ Micro SD với máy Linux ảo. Giả sử máy Linux ảo nhận diện thẻ Micro SD là tập tin dev/sdb. Ta tiến hành kết gắn thẻ Micro SD với thư mục /home/nhan/rootfs.

```
/$ mount dev/sdb /home/nhan/rootfs
```

- + Sau đó ta chép file Angstrom-km9260-01252010.tar.bz2 vào trong thư mục /home/nhan/rootfs.

- + Giải nén file này trong thư mục /home/nhan/rootfs

```
/$ tar -jxvf Angstrom-km9260-01252010.tar.bz2
```

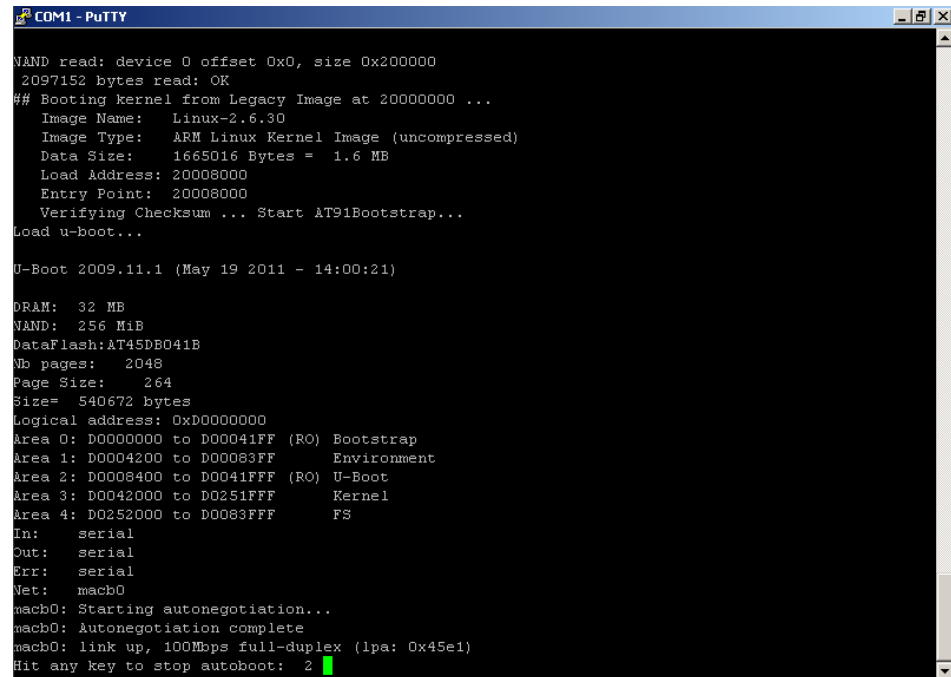
Vậy là ta đã cài đặt xong rootfs vào trong thẻ Micro SD. Chúng ta gắn thẻ Micro SD vào khe Micro SD trên KIT.

5. Cài đặt biến môi trường U-boot:

Sau khi cài đặt các file hệ thống vào các vùng nhớ tương ứng trên KIT thì ta kết nối KIT với máy tính bằng cổng COM và mở màn hình Terminal của KIT lên.

Nhấn nút Reset trên KIT để khởi động lại KIT.

Khi trên màn Terminal của KIT xuất hiện số đếm ngược như sau:



```
COM1 - PuTTY
NAND read: device 0 offset 0x0, size 0x200000
2097152 bytes read: OK
## Booting kernel from Legacy Image at 20000000 ...
Image Name:      Linux-2.6.30
Image Type:      ARM Linux Kernel Image (uncompressed)
Data Size:       1665016 Bytes = 1.6 MB
Load Address:    20008000
Entry Point:     20008000
Verifying Checksum ... Start AT91Bootstrap...
Load u-boot...

U-Boot 2009.11.1 (May 19 2011 - 14:00:21)

DRAM:  32 MB
NAND:  256 MiB
DataFlash: AT45DB041B
Nb pages:  2048
Page Size:  264
Size= 540672 bytes
Logical address: 0xD0000000
Area 0: D0000000 to D00041FF (RO) Bootstrap
Area 1: D0004200 to D00083FF Environment
Area 2: D0008400 to D0041FFF (RO) U-Boot
Area 3: D0042000 to D0251FFF Kernel
Area 4: D0252000 to D0083FFF FS
In:      serial
Out:     serial
Err:     serial
Net:     macb0
macb0: Starting autonegotiation...
macb0: Autonegotiation complete
macb0: link up, 100Mbps full-duplex (lpa: 0x45e1)
Hit any key to stop autoboot:  2
```

Hình 2.116: Màn hình Terminal khi đếm ngược từ 3 đến 0 để bắt đầu load U-boot vào SDRAM

Thì ta nhấn phím Enter để vào phần cài đặt của U-boot. Như đã trình bày phần trên thì u-boot có nhiều lệnh hỗ trợ và biến môi trường khác nhau. Trong phần này chúng ta sẽ cài đặt lại các biến môi trường của u-boot bằng các lệnh của u-boot.

- Nếu chúng ta nạp rootfs vào NAND Flash thì trong phần này ta cài đặt như sau:

```
U-boot> setenv bootcmd 'nand read 0x20000000 0x0 0x200000;bootm
0x20000000'
```

```
U-boot> setenv bootargs root=/dev/mtdblock1 rw rootfstype=jffs2
```

```
U-boot> save
```

***Chú ý các lệnh được nằm trong cùng một dòng.*

- Nếu chúng ta nạp rootfs vào thẻ Micro SD thì trong phần này ta cài đặt như sau:

VI. Các bước biên dịch Driver và cài đặt Driver:**1. Các bước biên dịch Driver:**

Để biên dịch một driver thì chúng ta phải tạo file makefile. Makefile giúp chúng ta biên dịch driver một cách nhanh chóng, tiện lợi.

Để tạo một file makefile thì chúng ta dùng một trình soạn thảo bất kỳ và soạn thảo chương trình makefile theo cấu trúc như sau:

```
export ARCH=arm
export CROSS_COMPILE=arm-none-linux-gnueabi-
obj-m = module.o
all:
    make -C /home/AT91SAM9260/KERNEL/linux-2.6.27 M=$(PWD)
    modules
clean:
    make -C /home/AT91SAM9260/KERNEL/linux-2.6.27 M=$(PWD) clean
```

Trong makefile trên ta có:

- Ở hàng thứ nhất và hàng thứ hai là chúng ta cài đặt giá trị cho hai biến ARCH và CROSS_COMPILE để sử dụng trình biên dịch chéo CROSS TOOLCHAINS.
- Hàng thứ ba, chúng ta khai báo driver mà ta muốn biên dịch. ở đây chúng ta biên dịch một driver có tên là module.
- Hàng thứ tư là nhãn thực thi lệnh biên dịch driver.
- Hàng thứ năm là lệnh make yêu cầu biên dịch driver. Lệnh make có cú pháp như sau:

+ make -C <đường dẫn đến source kernel> M=<đường dẫn đến thư mục chứa driver> <lệnh yêu cầu thực hiện>

Phía sau tùy chọn -C là thư mục chứa source kernel, tùy chọn M phải được gán cho đường dẫn đến thư mục chứa driver cần biên dịch. Lệnh yêu cầu thực hiện thường có hai lệnh sau:

modules: yêu cầu biên dịch driver đã được xác định.

clean: yêu cầu xóa các file đã biên dịch trước đó.

- Hàng thứ sáu là nhãn thực thi lệnh xóa các file đã biên dịch trước đó.
- Hàng thứ bảy là lệnh make yêu cầu xóa các file đã biên dịch trước đó.

Sau khi soạn xong chương trình makefile ta lưu dưới tên makefile. Sau đó ta chép makefile này và driver cần biên dịch vào trong một thư mục trong máy Linux ảo. Giả sử là thư mục /home/driver.

Sau đó ta mở màn hình terminal của máy Linux ảo lên, đi đến thư mục chứa makefile và driver cần biên dịch.

Ta cài đặt lại biến môi trường PATH để có thể sử dụng được chương trình biên dịch chéo CROSS-TOOLCHAINS.

Trong thư mục chứa makefile ta nhập dòng lệnh:

```
/home/driver$ make clean
```

```
/home/driver$ make all
```

Hai dòng lệnh trên là để thực hiện biên dịch driver bằng makefile. Sau khi biên dịch xong ta sẽ thấy trong thư mục chứa makefile sẽ có file *.ko. *.c là tên của driver mà chúng ta biên dịch. *.ko chính là driver mà chúng ta cài đặt vào hệ điều hành của KIT.

2. Các bước cài đặt Driver:

Chúng ta chép file *.ko mà ta vừa biên dịch vào KIT KM9260. Tại thư mục chứa file *.ko trên KIT chúng ta có thể dùng lệnh

```
$ insmod *.ko
```

Để cài đặt driver vào hệ điều hành trên KIT KM9260.

Ngoài ra, để tháo gỡ driver chúng ta có thể dùng lệnh:

```
$ rmmod *.ko
```

Để xem trên hệ điều hành có những driver nào đang được cài thì chúng ta có thể dùng lệnh:

```
$ lsmod
```

VII. Các bước biên dịch chương trình ứng dụng và chạy chương trình ứng dụng:

1. Các bước biên dịch chương trình ứng dụng:

Để biên dịch một chương trình ứng dụng thì chúng ta có hai cách là biên dịch trên máy Linux ảo và biên dịch trên KIT KM9260.

Biên dịch trên máy Linux ảo:

- Từ màn hình Terminal của máy Linux ảo chúng ta đi đến thư mục chứa chương trình ứng dụng cần biên dịch.
- Chúng ta cài đặt lại biến môi trường PATH sao cho sử dụng được trình biên dịch chéo CROSS-TOOLCHAINS.
- Sau đó ta nhập dòng lệnh sau:

```
$ arm-none-linux-gnueabi-gcc <tên chương trình cần biên dịch> -o <tên file muốn xuất ra>
```

- Sau khi biên dịch xong ta sẽ thấy file chương trình ứng dụng đã biên dịch có tên giống với tên file mà ta muốn xuất ra. Ta chép file này vào KIT và chạy chương trình.

Biên dịch trên KIT KM9260:

- Ta chép chương trình ứng dụng cần biên dịch vào KIT.
- Vào thư mục chứa chương trình cần biên dịch và nhập dòng lệnh sau để biên dịch chương trình:

```
$ gcc <tên chương trình cần biên dịch> -o <tên file muốn xuất ra>
```

- Sau khi biên dịch xong chúng ta sẽ thấy file xuất ra.

2. Chạy chương trình ứng dụng:

Chúng ta chép file ứng dụng vừa biên dịch ra vào trong kit KM9260. Tại thư mục chứa file ứng dụng trong kit ta chạy chương trình ứng dụng theo cú pháp sau:

```
$ ./<tên chương trình ứng dụng> <tham số thứ nhất> <tham số thứ hai> ...  
<tham số thứ n>
```

Trong đó:

`./` là cú pháp lệnh chạy một chương trình ứng dụng trong Linux.

`<tên chương trình ứng dụng>` là tên chương trình ứng dụng muốn chạy. Tên chương trình ứng dụng bắt buộc phải có và chính xác trong dòng lệnh. Nếu ta

nhập sai tên chương trình thì hệ điều hành sẽ báo lỗi không tìm thấy chương trình hoặc sẽ chạy một ứng dụng khác có tên mà ta vừa nhập vào.

<tham số thứ nhất> <tham số thứ hai> ... <tham số thứ n> là các tham số của chương trình ứng dụng. Tùy vào từng chương trình ứng dụng mà số lượng các tham số này sẽ khác nhau. Nếu chương trình ứng dụng không dùng đến các tham số này thì chúng ta có thể không cần sử dụng trong câu lệnh trên. Vai trò của các tham số này thì chúng ta sẽ hiểu rõ hơn trong phần lập trình nhúng nâng cao.

VIII. Tổng kết:

Kernel được viết bằng ngôn ngữ C. Nên chúng ta phải biên dịch thì mới nạp vào kit được. Kernel có thể chỉnh sửa, thêm chức năng bằng cách chỉnh sửa trong menuconfig.

Để hệ thống có thể hoạt động được thì chúng ta phải nạp các file hệ thống cho kit. Việc nạp các file hệ thống này cần phải làm chính xác từng bước. Tùy vào yêu cầu của hệ thống mà chúng ta cài đặt các thông số ban đầu phù hợp.

Các chương trình ứng dụng thì cần phải biên dịch bằng phần mềm biên dịch chéo thì mới có thể chạy trên kit nhúng được.

Như vậy, sau khi đọc xong phần này, người học đã được trang bị đầy đủ các kiến thức cho việc lập trình nhúng sau này. Các kiến thức này sẽ được ứng dụng rất nhiều trong quá trình sử dụng kit nhúng. Trong các phần sau, khi gặp các kiến thức này thì giáo trình sẽ không trình bày lại, mà chỉ gợi nhớ lại cho người học để người đọc tìm xem lại phần kiến thức đó.