بنام خدا

امتحان نهایی ساختار و زبان کامپیوتر ۴۰۱۲۶ ۱۳۹۴/۴/۲

مدت: دو ساعت

۲۷

الف - سؤالات درسی:

۱- سلسله مراتب حافظه چیست و چرا از چند نوع حافظه تشکیل شده است؟ (Memory hierarchy)

۲- مزیت و کاربرد نمایش متمم ۲ نسبت به نمایش متمم ۱ چیست؟

۳- تفاوت سرکشی (Polling) و وقفه ای (Interrupt) خطوط ورودی/خروجی چیست و بهتر است از هر کدام برای کدام نوع دسترسی I/O استفاده کرد؟

۴- فلسفه وجودی یا دلیل به کارگیری ماشین مجازی (Virtual machine) چیست؟

۵- چرا گوگل ماشین مجازی دالوریک را برای اندروید ارائه داد و جایگاه این ماشین نسبت به ماشین مجازی جاوا (JVM) در هنگام اجرای برنامه های تحت اندروید چیست؟

ب - سؤالات تشریحی

۶- فرض کنید یک فضای آدرس منطقی (= مجازی) برابر 2^8 بایت در اختیار یک برنامه یا فرایند (Process) می باشد و فضای واقعی حافظه از چهار صفحه (Page frame) ۳۲ بایتی تشکیل شده است. همین طور جدول صفحه (Page table) بدین شکل است:

Page	Frame #	Valid bit
0	2	1
1	-	0
2	-	0
3	0	1
4	1	1
5	-	0
6	-	0
7	3	1

۶-۱ مبادین (Fields) و طول آدرس مجازی را بیت با یک شکل نشان دهید.

۶-۲ مشخص کنید آدرس مجازی $0D_{hex}$ به کدام آدرس واقعی نگاشت می شود.

۶-۳ جدول صفحات داده شده را با یک شکل توصیف کنید (یعنی نشان دهید کدام صفحه حافظه واقعی به کدام صفحه حافظه مجازی نگاشت (map) شده است).

۶-۴ سیاست جایگزینی LRU را با یک مثال مراجعه به صفحات حافظه توضیح دهید.

صفحه ۱

۷ـ این بایت کد "جاوا" داده شده است:

Byte Code	Mnemonic	Description
03	iconst_0	Push integer 0 onto data stack
04	iconst_1	1
3C	istore_1	Remove integer from top of stack and place into local variable array at address 1
A7 x y	goto	skip number of bytes given in x,y
60	iadd	Add two integer operands that are at top of stack and push the sum
1B	iload_1	Load integer from local variable array position 1
A1 FF F5	if_icmplt	Compare two integer values at the top of stack for "less than" condition. If true add the following value to the program counter (FFF5 = -11 decimal)

⋮

۷ـ۱ با این دستورات و در صورت نیاز اضافه کردن منطق تعدادی الگوی دیگر یک حلقه جمع مثلاً ۱ تا ۱۰ پیاده‌سازی کنید.

۷ـ۲ کد باینری (بایت کد) را بنویسید.

۷ـ۳ مزیت این برنامه چیست؟

۸ـ۱ یک برنامه به زبان اسمبلی اینتل بنویسید که حافظه تصویر (مانیتور) را با حروف A، B، ... تا Z به طور تکراری پر کند. می‌توانید فرض کنید آدرس شروع این حافظه B800:0000 است و یک بایت خود کاراکتر است و بایت بعد ویژگی‌های آن (مثل خاکستری، چشمک‌زن) (توجه شود که تعداد تکرار متناسب با طول و عرض تصویر پیش‌فرض 40x80 کاراکتر باشد).

۸ـ۲ اگر بایت 00001111 شکل یک ویژگی رنگی باشد، می‌خواهید یکی در میان این رنگی شوند، چه تغییری در برنامه قبل لازم است؟

۹ـ دستور LDV R_j, V_i بردار V_i را در رجیستر R_j قرار می‌دهد (قرینه: Store: STV V_i, R_j). دستور Addv R_k, R_i, R_j دو بردار R_i و R_j را جمع می‌کند و در R_k می‌ریزد.

۹ـ۱ حلقه for i=0 to vectorlength-1 $V_3(i) = V_1(i) + V_2(i)$ را کدگذاری اسمبلی کنید.

۹ـ۲ دستورات ... SSE اینتل را نام ببرید ... ص ۲

موفق باشید

به نام خدا

تصحیح امتحان نهایی ساختار و زبان کامپیوتر ۱۲۶_۴۰ مورخ ۹۳/۴/۲

۱ – سلسله مراتب حافظه انواع تکنولوژی‌های حافظه را در بر می‌گیرد که سرعت دسترسی و حجم متفاوتی دارند (معمولاً هر چه سرعت بیشتر باشد گران‌تر است بنابراین کمتر و حساب شده‌تر در کامپیوتر استفاده می‌شود) ۲

+ حجم		سرعت دسترسی +
α 10GB–TB	Tape	α S
α 700MB–α GB	CD,DVD	α 0.1S
α TByte	Hard disk	α mS
α Gbit	Flash	α MS
α Gbit–Tbit	Main Memory(DRAM)	α 10 nS
α Kbit–Mbit	Cache (SRAM)	α nS
~ α bits	FF	α 100ps

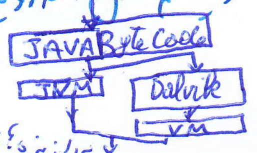
یک طراح کامپیوتر معمولاً از هر مرتبه تکنولوژی به اندازه مورد نیاز به کار می‌برد تا ترکیب مراتب مختلف حجم مناسبی از حافظه و میانگین سرعت دسترسی خوبی به سیستم حافظه در اجرای برنامه‌های مختلف برایش فراهم آورد.

۲ – ممیز ثابت دارای سادگی نمایش و پیاده‌سازی است زیرا تعداد ارقام اعشاری و در نتیجه دقت مطلق ثابت است در حالی که در ممیز شناور بسته به اینکه عدد کوچک یا بزرگ باشد ممکن است از ارقام اعشاری (دقت نمایش) بیشتر یا کمتر استفاده کنیم لذا پیاده‌سازی آن پیچیده‌تر و زمان‌برتر می‌باشد ولی دقت نسبی آن ثابت و برای اعداد کوچک‌تر دقت مطلق بهتری دارد. ۲
ممیز شناور امکان نمایش گستره وسیع‌تری را با اعداد محدود

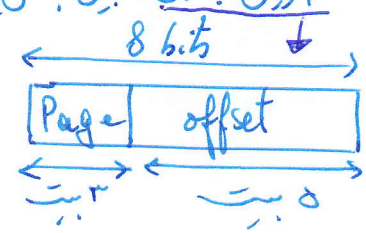
۳ – سرکشی یعنی اجرای برنامه ادواری (پریودیک) برای مراجعه به خطوط I/O برای خواندن و نوشتن دستگاه‌های جانبی به طور تکراری در حالی که وقفه دهی یعنی مراجعه به آن‌ها در صورت بروز وقفه یا درخواست و نه به طور متناوب و تکراری. ۲
روش سرکشی ساده‌تر برنامه نویسی و پیاده می‌شود ولی ممکن است نتایج تکراری و بی فایده‌ای تولید کند زیرا اتفاقی نیافتاده باشد بنابراین وقتی مناسب‌تر است که تغییرات I/O و یا تعداد آن‌ها زیاد است. وقفه دهی معمولاً در حالت موردی و اتفاقی بهتر است.

۴ – فلسفه وجودی ماشین مجازی امکان کار کردن با یک سیستم عامل یا ماشین (و ارتباطات خاص آن) در محیطی متفاوت (سیستم عامل دیگر یا پردازنده‌ای متفاوت) است. بدین ترتیب محیط پایه ادای ماشین مورد نظر را در بر دارد. همین طور نیازی به چندین برنامه exe نیست و مستقل از پردازنده برنامه تولید و توزیع می‌شود. ۲
در ضمن نقشه چند ماشین مجازی روی یک کامپیوتر را اجازه می‌دهد سرور چندین کاربر مختلف باشد.

۵ – ماشین مجازی دالویک به این دلیل توسط گوگل ارائه شد که حجم کمتری اشغال کند و سیستم عامل اندروید را برای برنامه‌های جاوا قابل استفاده کند. با اینکه جاوا توسط دالویک (ماشین مجازی) روی پردازنده‌های گوشی‌های موبایل یا تبلت اجرای شود ۲

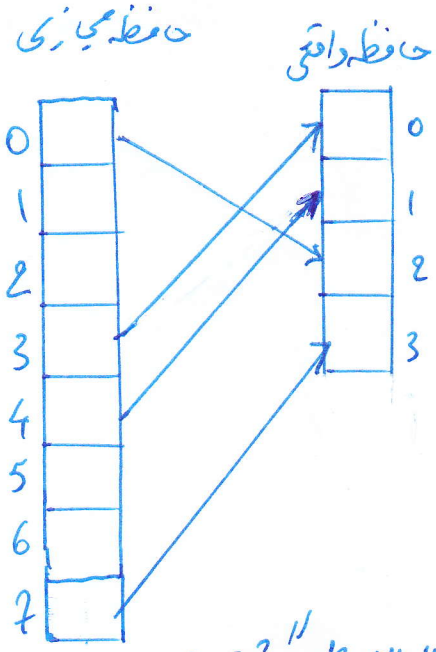


۶-۱ آدرس مجازی ۸ بیتی به شکل زیر است:



جدول صفحات (Page table):

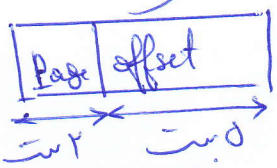
	Frame #	Valid bit
0	2	1
1	–	0
2	–	0
3	0	1
4	1	1
5	–	0
6	–	0
7	3	1



۶-۳ شکل نگاشت شماره صفحه مجازی به شماره صفحه واقعی

۲

آدرس واقعی



۶-۲ آدرس مجازی 0D hex برابر ۱۳ دسیمال

۱ و معادل باینری: 00001101 می‌باشد،

که به دو میدان 01101 و 000 تقسیم می‌شود.

page # offset

از آن جا که شماره صفحه ۰ حافظه مجازی هم اکنون در حافظه واقعی با شماره صفحه ۲ واقع است: حضور دارد و بیت اعتبار آن یک است) پس آدرس واقعی مربوط به 0Dh برابر 1001101 یا $4D_{hex}$ می‌شود یعنی آدرس دسیمال 77

۶-۴

صفحات مورد رجوع مجازی ←

صفحات واقعی ←

مراجعات: 0 3 0 3 4 0 7 3 6 5 1 7 2 4 5

2 0 2 0 1 2 3 0 1 2 3 0 1 2 3

جایگزینی صفحه قبلی با کنونی - دلیل الگوریتم LRU

۲

در الگوریتم LRU صفحه یا بلوکی را جایگزین می‌کنیم که دورتر نسبت به بقیه نیاز باشد (یعنی اخیراً مورد مراجعه قرار نگرفته باشد و این اتفاق هنگام خواندن صفحه مجازی ۴ می‌افتد که می‌خواهد وارد حافظه واقعی شود ولی فضا پر است ولذا صفحه مجازی ۲ را بیرون می‌اندازد (روی دیسک می‌نویسد) و خود به جای آن در صفحه واقعی ۱ قرار می‌گیرد.

۷- رجوع به برگه پیوست.

(۴) مزیت برنامه Java Byte code در این است که می‌تواند توسط ماشین مجازی JVM روی هر سیستم رایانه‌ای اجرا شود ولذا کدهای اولیه قابل حمل روی بستهای (Platforms) مختلف می‌باشد.

سؤال ۷ :

PC	Source Line #	
	1	public class Simple {
	2	public static void main (String[
00	3	int i = 0;
02	4	double j = 0;
04	5	while (i < 10) {
07	6	i = i + 1;
0B	7	j = j + 1.0;
0F	8	} // while
15	9	} // main()
	A	} // Simple()

اقتباس از کتاب :
L. Null, J. Lobur, " Computer organization and architecture", Jones & Bartlett Learning 3rd Edition, 2012.

Offset (PC value)	Bytecode	Mnemonic	Meaning
			Variable initialization
0	03	iconst_0	Push integer 0 onto data stack.
1	3C	istore_1	Remove integer from top of stack and place into local variable array at address 1.
2,3	0E, 49	dconst_0	Push double precision constant 0 onto the stack. → store double constant 0 into local variable at 0=2
4	A7	goto	Skip number of bytes as given in following two bytes. This means we'll skip next 11 bytes. This means add 0B to the program counter.
5	00		
6	0B		
			Loop body
7	1B	iload_1	Push an integer value from local array onto stack.
8	04	iconst_1	Push integer constant 1.
9	6C	iadd	Add two integer operands that are at top of stack. (Sum is pushed.)
A	3C	istore_1	Remove integer from top of stack and place into local variable array at address 1.
B	28	dload_2	Load double variable from local array and place it on the stack.
C	0F	dconst_1	Push double precision constant 1 onto the stack.
D	63	dadd	Add the double precision values that are at the top of the stack.
E	49	dstore_2	Store the double sum from the top of the stack to local variable array position 2.
			Loop condition
F	1B	iload_1	Load integer from local variable at... position 1
10	10	bipush	Push the following byte value onto the stack. (10 decimal)
11	0A		
12	A1	if_icmplt	Compare two integer values at the top of stack for "less than" condition. If true (i < 10), add the following value to the program counter. Note: FFF5 = -11 decimal.
13	FF		
14	F5		
15	B1	return	otherwise, return

PC
+4
-11
PC

-۸

```
VideoDisp   dw   B800h      ; آدرس شروع نقطه تصویر
ChCode1     db   0          ; کد نمایش معمولی
chcode2     db   0FFh       ; کد نمایش رنگی
Rows        db   40         ; تعداد ردیف های صفحه نمایش
Columns     db   80         ; " ستون های " "
mov         Ax, VideoDisp
mov         DS, Ax
mov         Cx, Rows        ; شروع حلقه نمایش ردیف به ردیف
RowDisplay:
 Push       Cx
 mov        Cx, Columns     ; شروع حلقه نمایش ستون به ستون داخل هر ردیف.
 mov        al, 'A'
 mov        ah, chcode1
ColDisplay:
 mov        ES:[Bx], Ax     ; نمایش کاراکتر با ویژگی رنگی آن
 inc        Al              ; آماده کردن حرف بعدی برای نمایش
 cmp        Al, 'Z'         ; اگر از 'Z' بزرگتر شد دوباره از
 Jle        skip1           ; 'A' شروع به نمایش کن
 mov        Al, 'A'
skip1: inc  Bx
 inc        Bx
 loop       ColDisplay
 Pop        Cx
 loop       RowDisplay
```

۲

۸-۲ اگر قرار باشد یکی در میان کد 0 و سپس 0F بعد از هر حرف ارسال شود کافی است این دستورات را به بدنه ColDisplay اضافه کنیم:

```
ColDisplay:  mov  ES:[Bx], Ax
             cmp  Ah, ChCode1
             Beq  Toggle1
             mov  Ah, ChCode1
             Jmp  GoAhead
Toggle1:     mov  Ah, chCode2
GoAhead:     inc  Al
             :
```

بدین ترتیب که اگر کد 0 بود آن را 0F و اگر 0F بود 0 می کنیم.

۱

۹-۱

```
LdV   R1, V1
LdV   R2, V2
addV  R3, R1, R2
```

۲

۹-۲ دستورات اینتل از نوع SIMD (اینتل SSE) شامل: انتقال بردار مثل movAPS، movUPS، حسابی مثل AddPS، MulPS، مقایسه CMPPS، shuffle مثل ShufPS و تبدیل نوع CVTPI2PS و عملیات بیتی مثل AndPS هستند.

۲

۲۶۰۹